Notas de Aula
Prof. Adriano Milanez

Programação Web I
Responsive Web Development
Front-End Design Engineering
Programação Web Responsiva

[aula 15]
[Atualizado em: 14/08/2025]

[VITE – FrameWorks – Introdução]

O QUE SÃO FRONT-END FRAMEWORKS?

O QUE É UM FRAMEWORK?

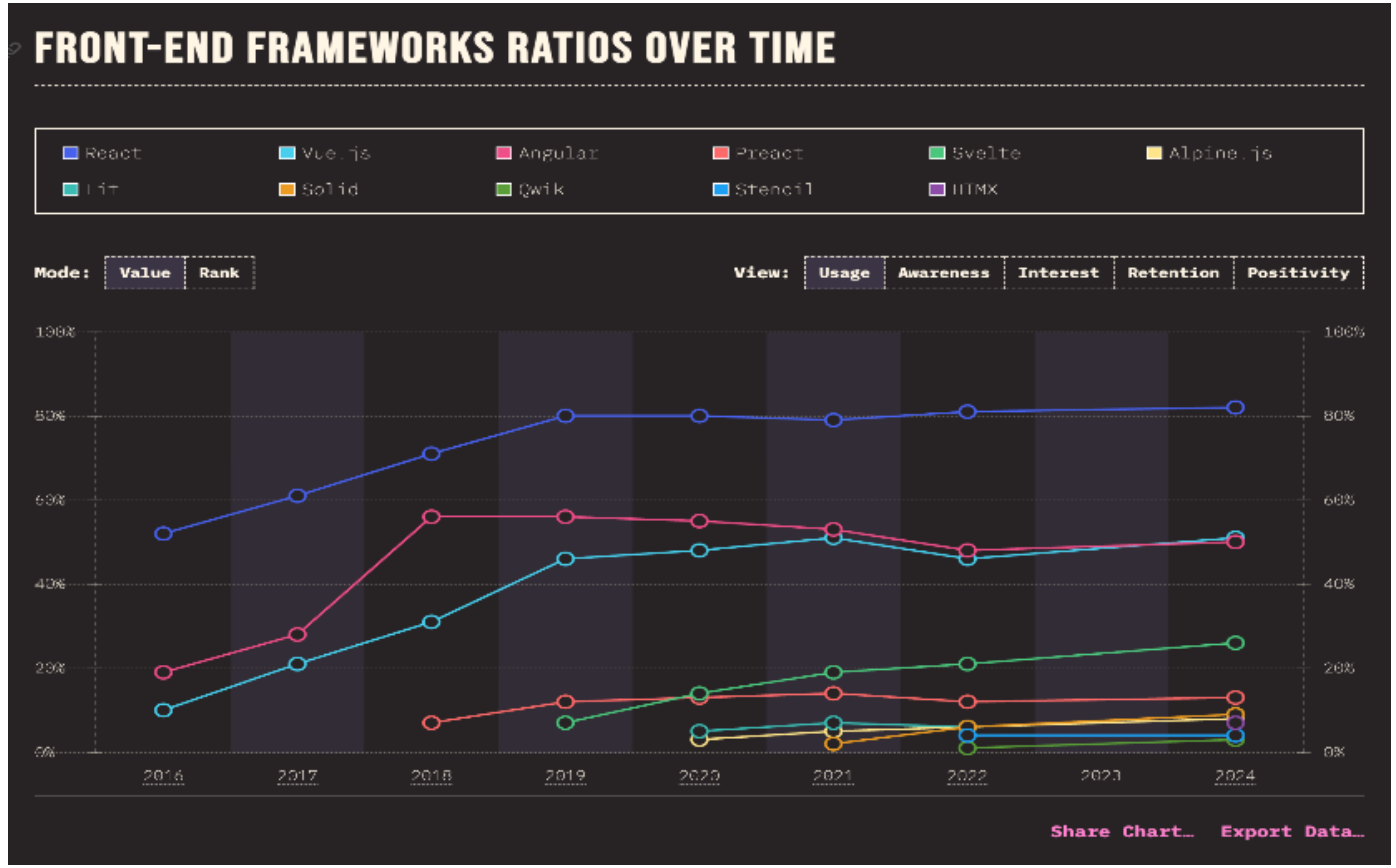
- ✓ São conjuntos de ferramentas e bibliotecas que facilitam o desenvolvimento de interfaces de usuário para websites, aplicações web e/ou mobile.
- ✓ Eles fornecem uma estrutura organizada, componentes prontos para uso e estilos pré-definidos, o que acelera o desenvolvimento e garante consistência visual.
- ✓ Além disso, oferecem suporte para responsividade e integração com bibliotecas de JavaScript para adicionar interatividade.
- ✓ É basicamente um grande driver que permite que os desenvolvedores personalizem os componentes predefinidos de um aplicativo de acordo com a necessidade do aplicativo. Também ajuda a melhorar a velocidade geral de um aplicativo.

O QUE É UM FRAMEWORK?

- ✓ **React**: Desenvolvido pelo Facebook, é usado para construir interfaces de usuário interativas com componentes reutilizáveis e gestão eficiente do estado.
- ✓ **Angular**: Desenvolvido pelo Google, é um framework completo que permite criar aplicações web dinâmicas e escaláveis com uma abordagem estruturada.
- ✓ **Vue.js**: Focado na simplicidade e flexibilidade, Vue.js facilita a criação de interfaces interativas e é conhecido por sua curva de aprendizado acessível.
- ✓ **Bootstrap**: Um framework CSS que fornece uma ampla gama de componentes e estilos pré-definidos para criar designs responsivos e modernos rapidamente.
- ✓ **Tailwind CSS**: Um framework CSS utilitário que permite construir interfaces de usuário personalizadas com uma abordagem baseada em classes utilitárias.
- ✓ **Svelte**: Um framework inovador que compila componentes em código altamente eficiente, oferecendo uma experiência de desenvolvimento mais fluida e rápida.

Não existe um melhor, mas sim aqueles que melhor se adequa a sua necessidade.

O QUE É UM FRAMEWORK?



Fonte: https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/, acesso em 03 de julho de 2025.

O QUE É UM FRAMEWORK?

- ✓ O **React** é uma biblioteca Javascript criada pelo Facebook, utilizada para criar interfaces para usuários, com uma particularidade de renderizar somente a parte da tela que é necessária, aumentando assim em muito a performance da página.

- ✓ É de código aberto e quem mantém e evolui o **React** é você, ou seja, a comunidade.

- ✓ Seu foco principal é transformar a experiência do usuário mais eficiente, tornando a aplicação mais leve e performática, permitindo a **reusabilidade** de componentes.

- ✓ É uma biblioteca da linguagem **JavaScript** e que atua na camada **View** de um projeto com MVC.

- ✓ Lembrando que não trabalharemos com **Javascript** e sim com **TYPESCRIPT**.

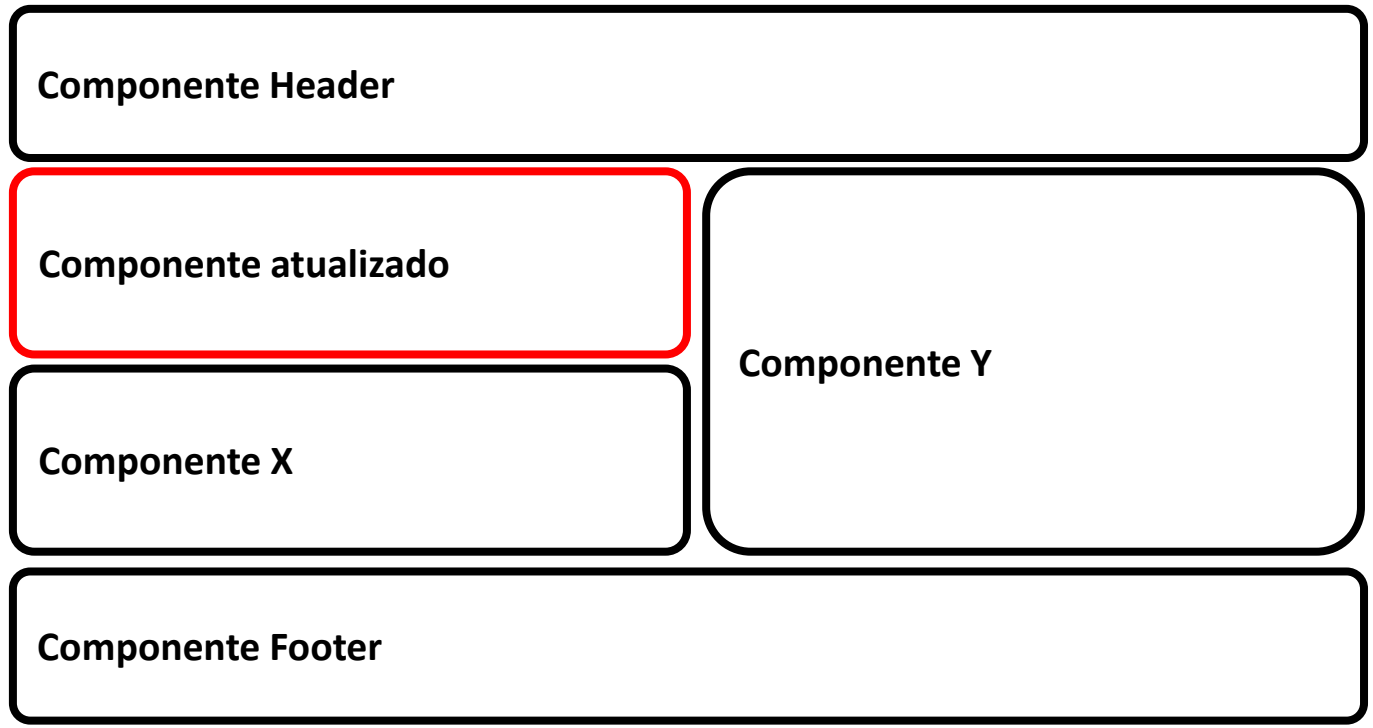
O QUE É UM FRAMEWORK?

- ✓ No **React** tudo é **Javascript**, até os elementos HTML são criados por ele através do JSX, uma extensão de sintaxe que nos permite trazer a criação dos elementos HTML para dentro do Javascript.

- ✓ Para que as mudanças dos componentes da tela sejam harmoniosas, ele utiliza o Virtual DOM (VDOM) que gerencia os componentes em memória e sincroniza com o DOM real, utilizando a biblioteca do ReactDOM. Isso aumenta a performance, melhorando até a classificação nos motores de busca.
  - ✓ DOM – Manipulação dos dados HTML em tempo real com o React
  - ✓ É possível alterar qualquer texto das tags do HTML.
  - ✓ Os atributos do HTML, também podem ser alterados pelo React.
  - ✓ Possibilidade de remover e adicionar as tags do HTML, através do React.
  - ✓ Tem o mesmo nível de ação no HTML e CSS.
  - ✓ O React é muito utilizado com o HTML, através dos eventos, um exemplo muito conhecido é o onclick do HTML.

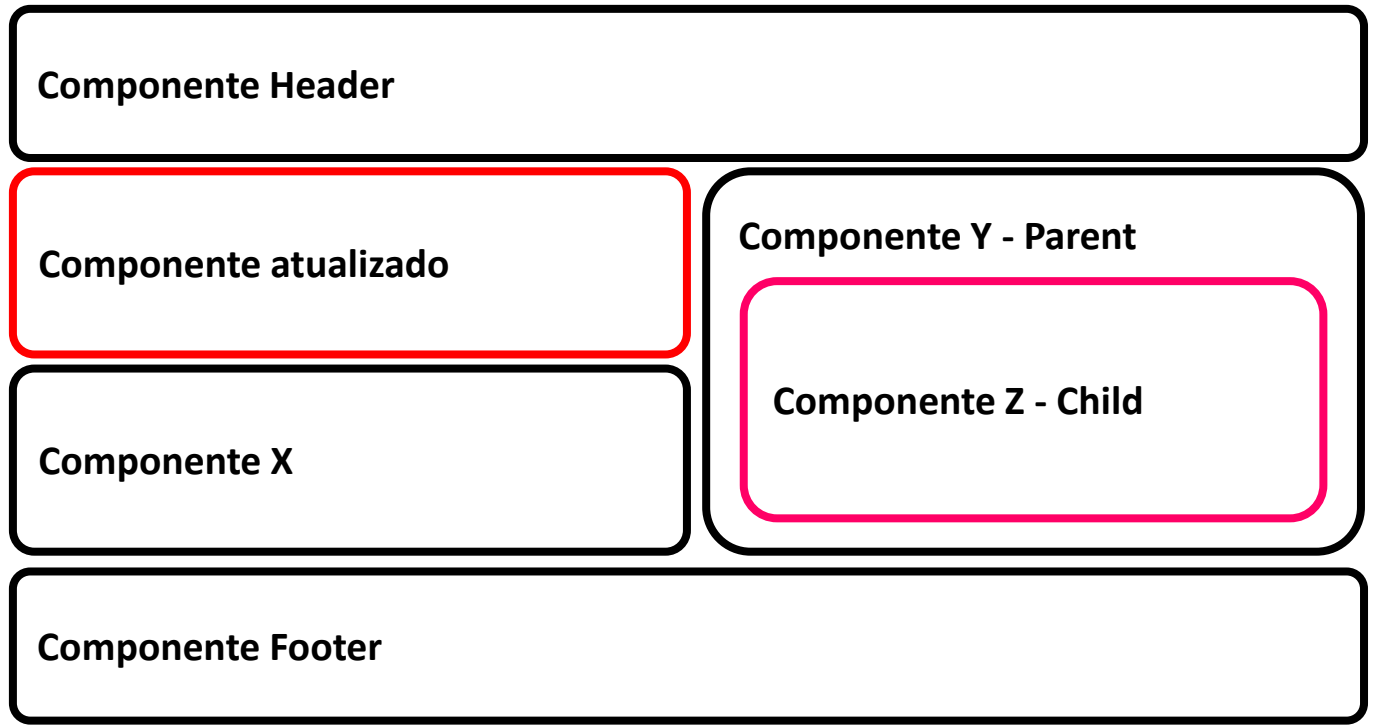
O QUE É UM FRAMEWORK?

- ✓ O **React**, por trabalhar com componentes, só precisa carregar a parte da página que foi alterada, mantendo as demais partes, deixando o trabalho mais rápido.



O QUE É UM FRAMEWORK?

- ✓ Ele trabalha com um fluxo unidirecional, em um único sentido, ou “One-way data flow”. As informações devem sempre vir do elemento pai (Parent) para o elemento filho (Child).



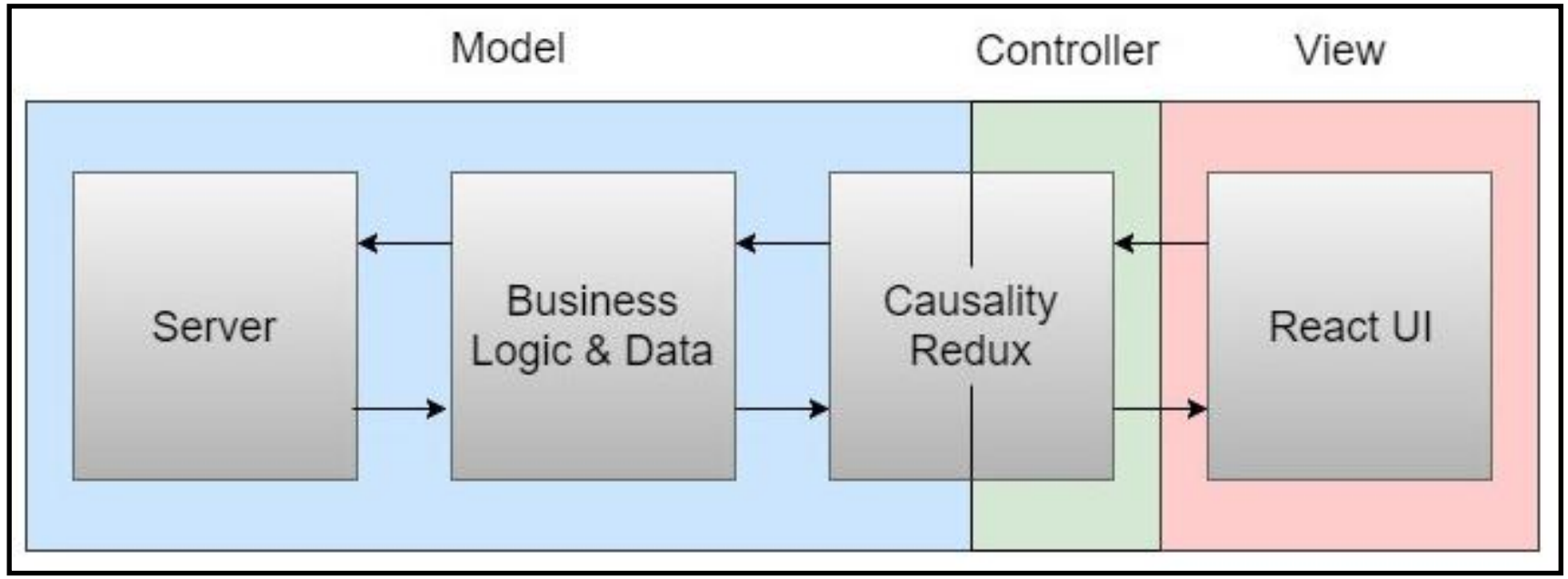
O QUE É UM FRAMEWORK?

- ✓ **Pontos positivos**
  - ✓ Muita eficiência na utilização dinâmica entre o JS e o HTML, refletindo no interface para o usuário (UI).
  - ✓ Sabendo a sintaxe e lógica do JS, torna fácil a utilização do React.
  - ✓ Mais entrega com menos esforço para o UI.

- ✓ **Pontos negativos**
  - ✓ Trabalha apenas na camada da View.
  - ✓ Para a utilização é necessário conhecimento prévio em JS, na sintaxe, lógica de programação, HTML e CSS.

O QUE É UM FRAMEWORK?

- ✓ Um exemplo é por exemplo na UI, se tem um botão que ao ser clicado envia para um *controller* (lembrando do conceito de JSP podemos lembrar do *package servlet*) que então aciona o código backend da aplicação.



VITE – INTRODUÇÃO

O QUE É VITE?

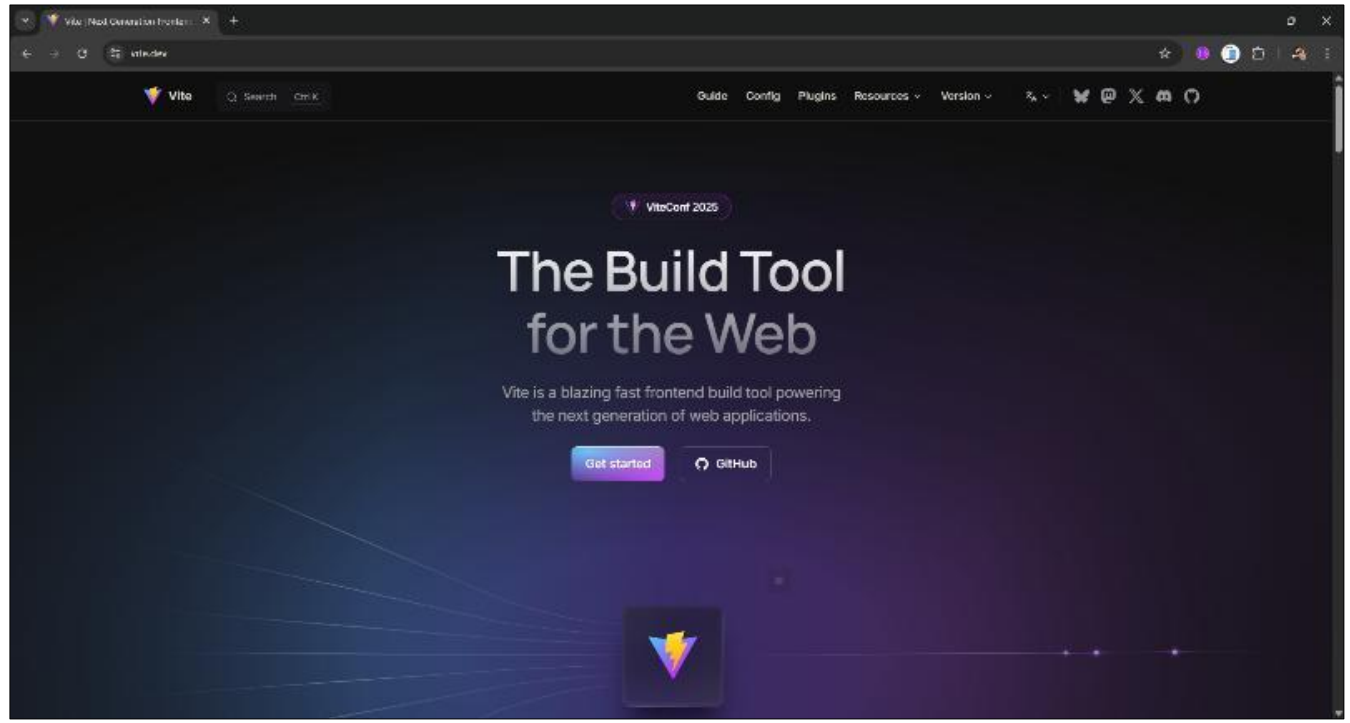
- ✓ **Vite** (do francês *vite*, que significa "rápido", pronunciada /vit/, como "veet") **é uma ferramenta de construção que visa proporcionar uma experiência de desenvolvimento mais rápida e enxuta para projetos web modernos.**

- ✓ Projetada para oferecer uma experiência de desenvolvimento **rápida**, **moderna** e **eficiente**.

- ✓ Criada por **Evan You** (o mesmo criador do Vue.js)

O QUE É VITE?

- ✓ **Vite** é uma ferramenta moderna de *build* e desenvolvimento Front-End.
- ✓ Tecnicamente, **Vite** é um **dev server** que acelera o desenvolvimento, **build** de aplicações *web* modernas e **bundler**.



O QUE É VITE?

- **Dev Server**
  - Quando você roda *Vite* ou npm run dev, ele:
    - Usa ES Modules nativos do navegador para carregar os arquivos diretamente.
    - Serve os arquivos rapidamente.
    - Aplica Hot Module Replacement (HMR) instantâneo quando você altera algo (sem recarregar toda a página).

- **Build Tool**
  - Quando você roda vite build, ele:
  - Usa o Rollup (internamente) para empacotar os arquivos para produção.
  - Faz tree-shaking, minificação, code splitting, etc.
  - Gera uma build leve, eficiente e pronta para produção.

O QUE É VITE?

- **Bundler**
  - Aceita JavaScript, TypeScript, CSS, SCSS, PostCSS, Vue, React (com plugin), Svelte, etc.
  - Suporta recursos modernos como:
    - Importações dinâmicas
    - Carregamento assíncrono
    - Plugins personalizados
    - Suporte a monorepos com pnpm, yarn ou npm

O QUE É VITE?

O VITE...

É...	Não é...
Um **bundler** moderno	Um framework
Um **servidor de dev** rápido	Um meta-framework
Base para frameworks modernos	Algo com roteamento, SSR, etc.
Substituto do Webpack/Parcel	Algo como VITE ou Nuxt

O QUE É VITE?

- **O que é um bundler?**
  - Um **bundler** (em português, “empacotador”) é uma **ferramenta que pega todos os arquivos do seu projeto (JavaScript, CSS, imagens, etc.) e os agrupa em um ou mais arquivos otimizados para serem executados no navegador**.

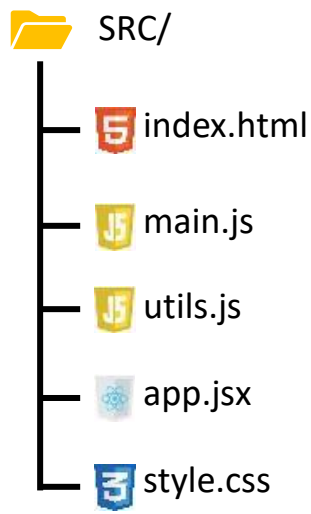
O QUE É VITE?

- **Por que precisamos de um bundler?**
  - Imagine que seu projeto tem:
    - main.js
    - utils.js
    - app.jsx
    - style.css
    - logo.png
  - O navegador não entende import de JS moderno, JSX, SCSS ou outras linguagens diretamente.
  - Carregar muitos arquivos separadamente pode ser lento (várias requisições HTTP).
  - E se você usar módulos, precisa de algo que combine tudo em um arquivo executável.
  - Aí entra o **bundler**.

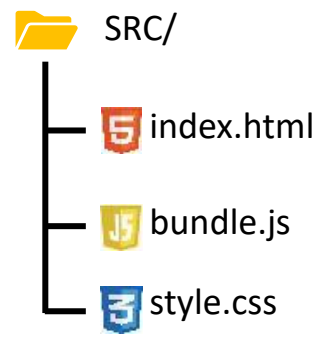
O QUE É VITE?

- ✓ **O que ele faz?**
  - ✓ Lê sua árvore de dependências (quem importa quem)
  - ✓ Compila (ex: transpila JSX ou TypeScript para JS)
  - ✓ Agrupa arquivos (code splitting ou em um único bundle)
  - ✓ Minifica (remove espaços, reduz nomes de variáveis)
  - ✓ Otimiza para produção

Antes do bundler



Depois do bundler



← tudo compilado e agrupado aqui

O QUE É VITE?

- ✓ Em resumo,
  - ✓ um **bundler** transforma o seu código moderno em um **conjunto de arquivos otimizados para rodar em qualquer navegador**, cuidando de **empacotar, compilar e otimizar** tudo que seu app precisa.

O QUE É VITE?

- **Características**
  - **Desenvolvimento ultra-rápido com ES Modules (ESM)**
    - Usa nativamente os ES Modules no navegador durante o desenvolvimento.
    - Não precisa empacotar tudo antes de rodar o projeto localmente, o que acelera muito o tempo de inicialização.
  - **Hot Module Replacement (HMR) eficiente**
    - Atualizações instantâneas na interface enquanto você desenvolve, sem precisar recarregar toda a página.
  - **Build otimizado com Rollup**
    - Para produção, o Vite usa o Rollup como bundler, garantindo builds otimizadas e pequenas.

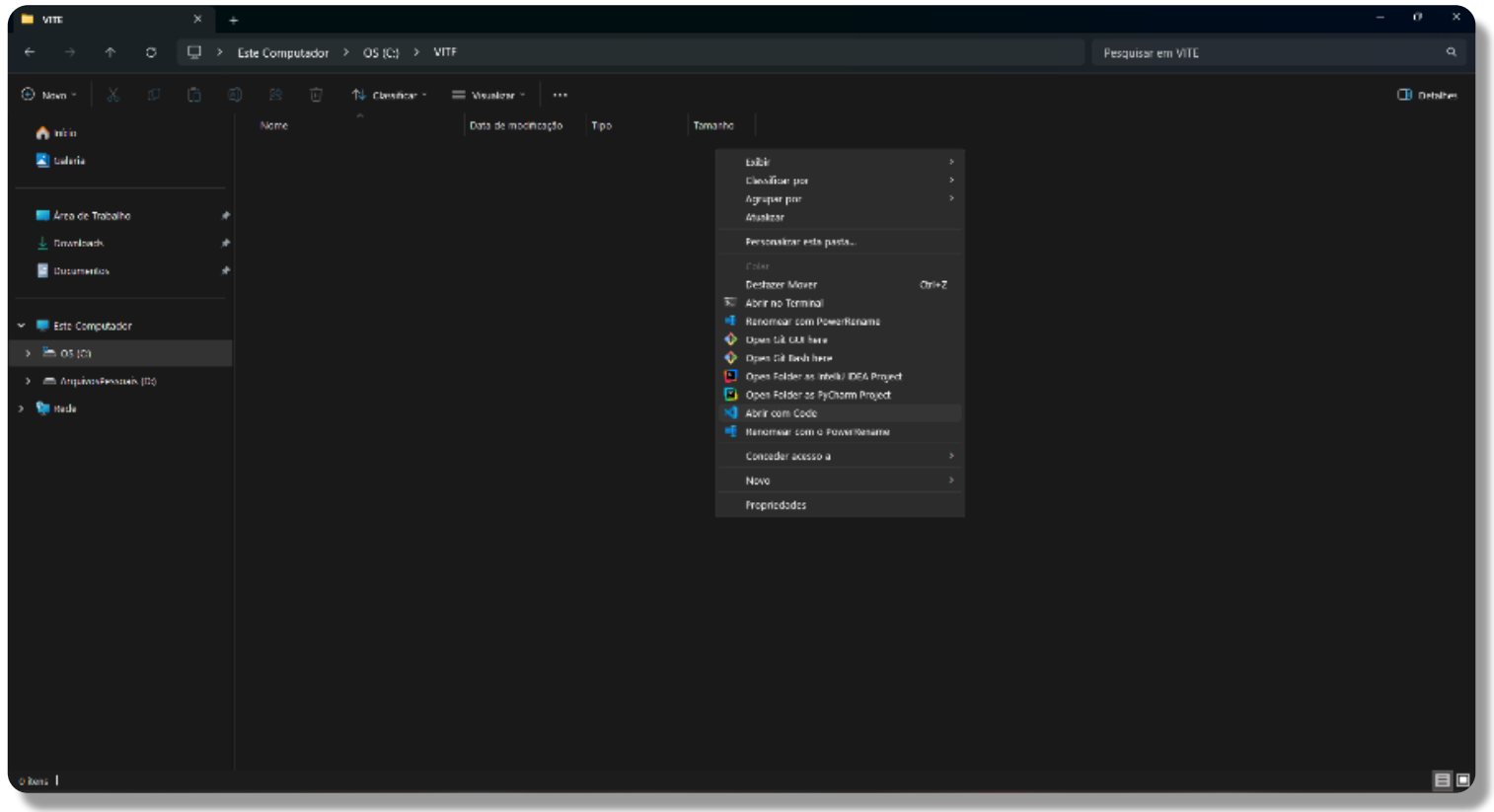
O QUE É VITE?

- **Características**
  - **Suporte a frameworks**
    - Suporte nativo para Vue.js, React, Svelte, Lit, Preact, entre outros.
    - Basta instalar o plugin correto, como @vitejs/plugin-react.
  - **Configuração simples**
    - Um projeto pode ser iniciado rapidamente com comandos como:
      - `npm create vite@latest`
  - **TypeScript, JSX, CSS, PostCSS, Tailwind, etc.**
    - Suporte direto ou com mínima configuração para essas tecnologias modernas.

VITE – CRIANDO A PRIMEIRA APLICAÇÃO

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

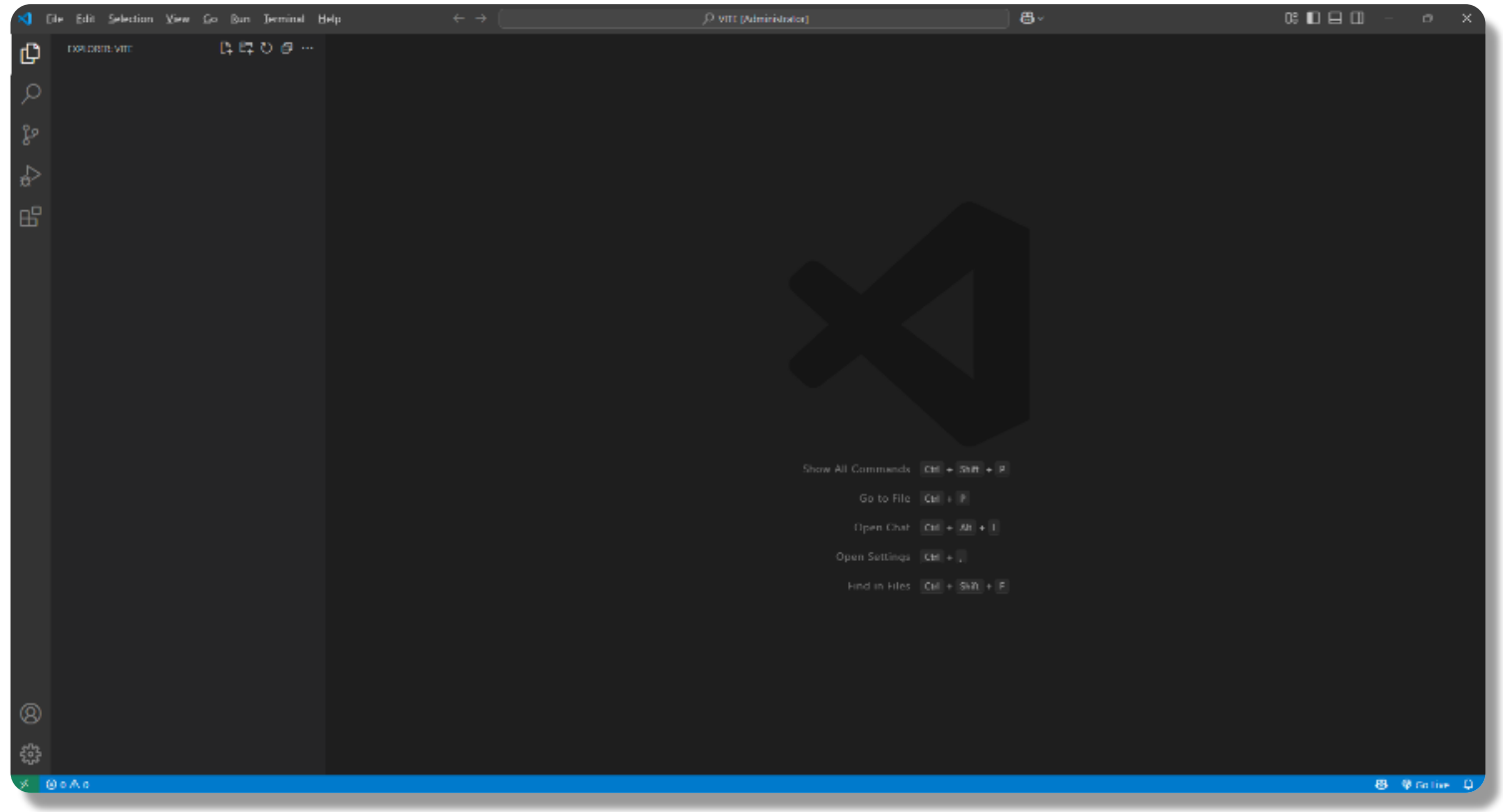
- **Passos iniciais**
  - Em seu Gerenciador de Arquivos crie uma pasta chamada “VITE” em seguida, clique com o botão direito e selecione “Abrir com o Code”.



Criaremos uma pasta chamada VITE na raiz e apontaremos todas as aulas/projetos para esse diretório. Para cada aula/projeto manteremos o padrão: **aula01**, **aula02**, **aula02a**, **aula03b** e assim sucessivamente.

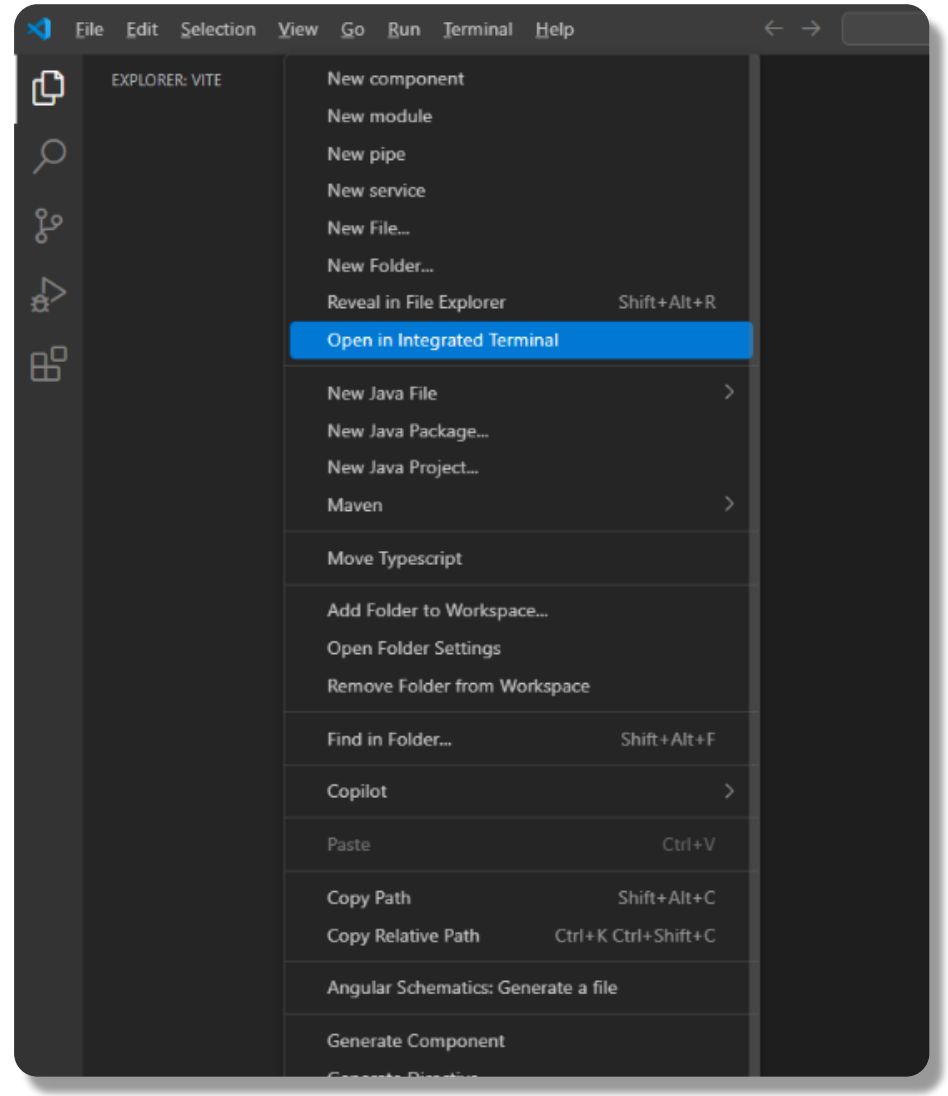
VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- **Passos iniciais**
  - Possivelmente seu VSCode deve ter aberto desta forma



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**
  - ✓ Possivelmente seu VSCode deve ter aberto desta forma
  - ✓ Na sequência clique o botão direito do mouse e selecione “Open in Integrated Terminal”

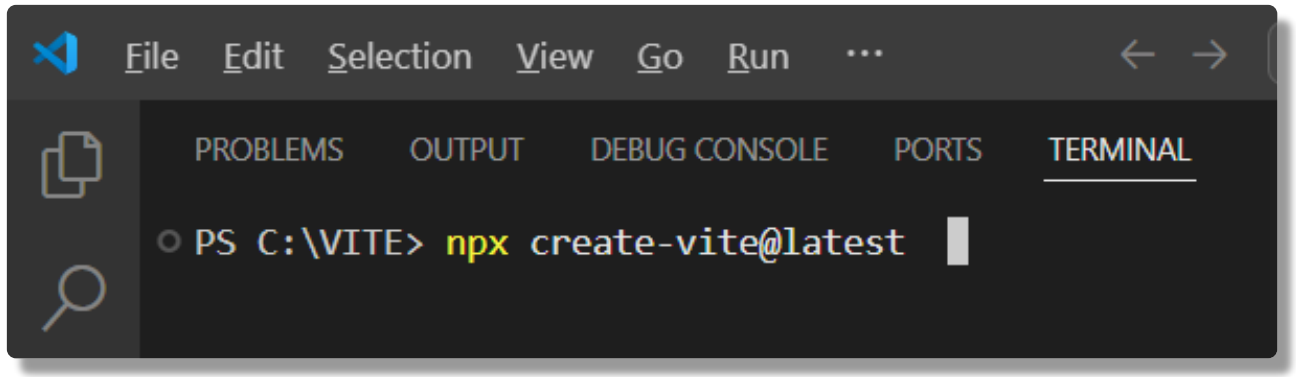


VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:

   - ✓ `npx create-vite@latest`



CASO ERRO DE SEGURANÇA:
Abra o seu **Powershell** como **administrador**
Execute o comando
`Set-ExecutionPolicy RemoteSigned`
Confirme a alteração digitando **A** no terminal e aperte a tecla Enter
Feche o Powershell
Execute novamente o Powershell, dessa vez sem modo administrador
Execute o comando
`node --version`
Se aparecer a versão do node, o problema foi corrigido com sucesso.

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- **Passos iniciais**
  - Vamos analisar palavra por palavra a instrução: **`npx create-vite@latest`**
    - A instrução **`npx create-vite@latest`** executa a última versão do pacote create-vite sem precisar instalá-lo globalmente.
    - Na prática, você roda o instalador oficial do Vite para criar um projeto do zero com a versão mais atual da ferramenta.

  - **NPX**
    - **O que é**: Um utilitário do Node.js que vem junto com o NPM (Node Package Manager) a partir da versão 5.2. Ele foi criado para facilitar a execução de pacotes diretamente, sem precisar instalá-los globalmente na máquina.
    - **O que significa**: **N**ode.js **P**ackage e**X**ecutor - Ele executa pacotes do NPM (Node.js) diretamente da linha de comando.
    - **Função**: Executa pacotes NPM diretamente sem a necessidade de instalá-los globalmente na máquina.
    - **Por que usar aq**ui: Permite rodar o comando create-vite sem precisar instalá-lo previamente.

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- **Passos iniciais**
  - Vamos analisar palavra por palavra a instrução: **`npx create-vite@latest`**
  - **CREATE-VITE**
    - **O que é**: É o **nome do pacote npm** que você quer executar..
    - Esse pacote é o *scaffolder* oficial do Vite, responsável por criar um novo projeto Vite configurado do zero.
    - Ele gera a estrutura inicial do projeto, com arquivos e dependências básicas.

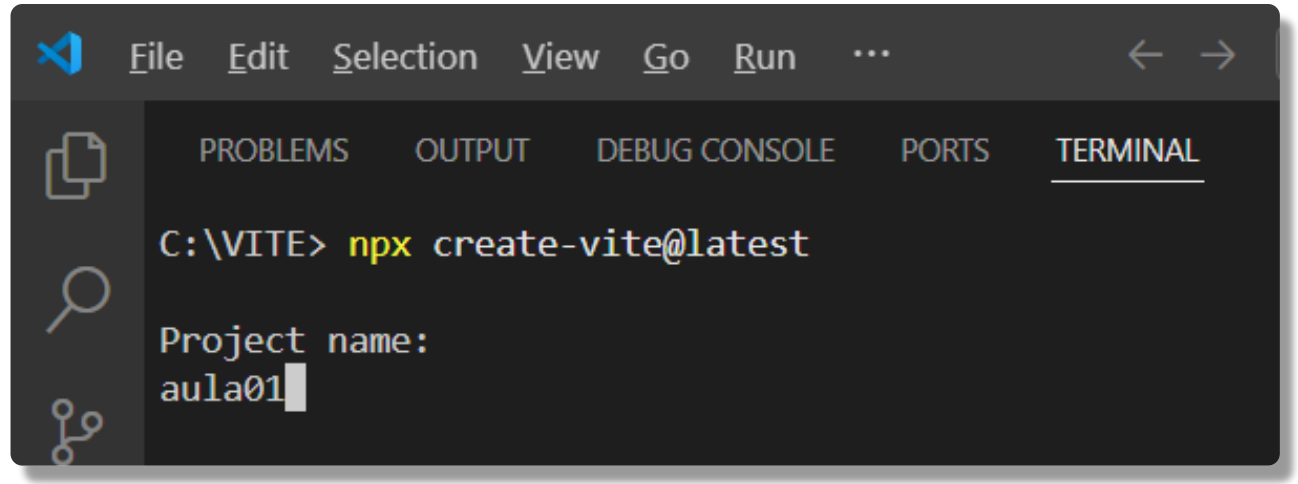
VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- **Passos iniciais**
  - Vamos analisar palavra por palavra a instrução: **`npx create-vite@latest`**
  - **@LATEST**
    - É uma *tag* de versão do pacote.
    - Significa que você quer usar a última versão publicada no npm do pacote create-vite.
    - Sem isso, o npm pode usar uma versão instalada localmente, cacheada ou padrão.
    - Isso garante que você está sempre usando a versão mais recente e com correções/funções atualizadas.

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
   - ✓ `aula01`



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

1. **Somente letras minúsculas, números, hífens (-) e underlines (_)**
   - ✓ **Exemplos corretos**
     - ✓ meu-projeto
     - ✓ front-musics-app
     - ✓ mini_loja
   - ✓ **Exemplos incorretos**
     - ✓ MeuProjeto (letras maiúsculas são proibidas para nomes de pacote npm)
     - ✓ meu*projeto! (caracteres especiais proibidos)

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

1. **Somente letras minúsculas, números, hífens (-) e underlines (_)**

2. **Sem espaços**
   - ✓ **Exemplos corretos**
     - ✓ meu-projeto-react (**kebab case**)
     - ✓ meu_projeto_react (**snake case**)
   - ✓ **Exemplos incorretos**
     - ✓ meu projeto

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

1. **Somente letras minúsculas, números, hífens (-) e underlines (_)**
2. **Sem espaços**

3. **Não pode começar com número, hífens (-) ou underline (_)**
   - ✓ **Exemplos corretos**
     - ✓ app123
     - ✓ meu-app-123
   - ✓ **Exemplos incorretos**
     - ✓ 123projeto
     - ✓ -meuapp
     - ✓ _meuapp

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

1. **Somente letras minúsculas, números, hífens (-) e underlines (_)**
2. **Sem espaços**
3. **Não pode começar com número, hífens (-) ou underline (_)**
4. **Não usar nomes reservados ou existentes no npm**
   - ✓ Nomes como react, vite, express, etc. já estão registrados.
   - ✓ Se tentar usar, o npm install pode dar erro ou avisos.

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

1. **Somente letras minúsculas, números, hífens (-) e underlines (_)**
2. **Sem espaços**
3. **Não pode começar com número, hífens (-) ou underline (_)**
4. **Não usar nomes reservados ou existentes no npm**

Dica de boa prática

- ✓ Use letras minúsculas + hífens para separar palavras.
- ✓ **Exemplo ideal**: catalogo-miniaturas-vite-react

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

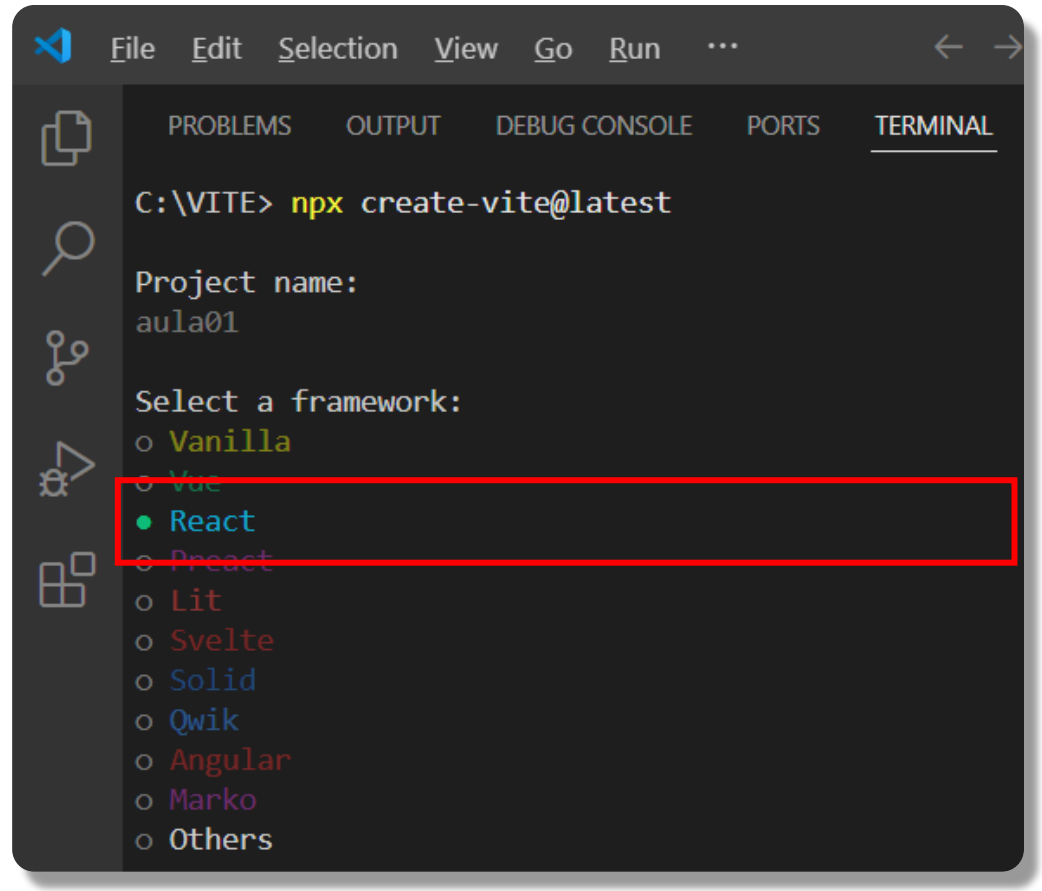
✓ **Regras de nomenclatura para nomes de projeto (pasta e pacote npm)**

Regras	Permitido?	Exemplo
Letras minúsculas	✅	meu-app
Números (desde que não iniciem)	✅	app123
Hífens (-)	✅	projeto-teste-vite
Underlines (_)	✅	meu_app_teste
Letras maiúsculas	❌	MeuProjeto
Espaços	❌	meu projeto
Caracteres especiais (!@#)	❌	meu@projeto!
Começar com número ou traço	❌	123app, -projeto

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

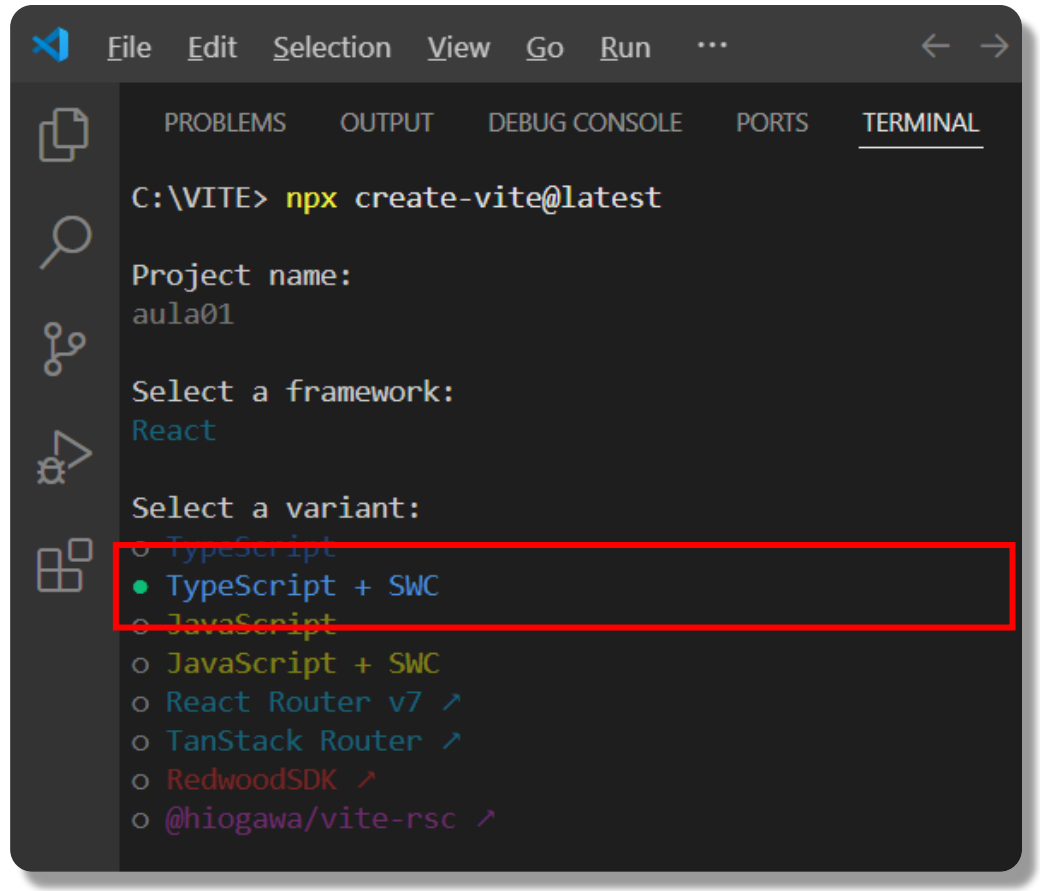
1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
3. Vamos selecionar o *framework* que iremos criar o nosso projeto.
   - ✓ Vamos selecionar, com a seta de direção, o **React**



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
3. Vamos selecionar o *framework* que iremos criar o nosso projeto.
4. Agora vamos selecionar a 'linguagem' que iremos trabalhar.
   - ✓ Vamos selecionar, com a seta de direção, o **TypeScript + SWC**



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- **Qual a diferença entre TypeScript e TypeScript + SWC?**
  - **TypeScript**
    - Verifica os tipos
    - **Transpila**[1] seu código .ts ou .tsx para .js padrão
    - É o mais preciso em relação à linguagem
    - Mais lento (especialmente em projetos grandes)
  - **TypeScript + SWC (Speed Web Compiler)**
    - **Transpila** o código rapidamente (escrito em Rust, a documentação afirma ser "20x mais rápido que o Babel em um único thread e 70x mais rápido em quatro núcleos".)
    - Não faz verificação completa de tipos (somente strip dos tipos)
    - Ideal para ambientes de desenvolvimento ou produção que já fazem o type checking separadamente (como via tsc --noEmit ou IDE)

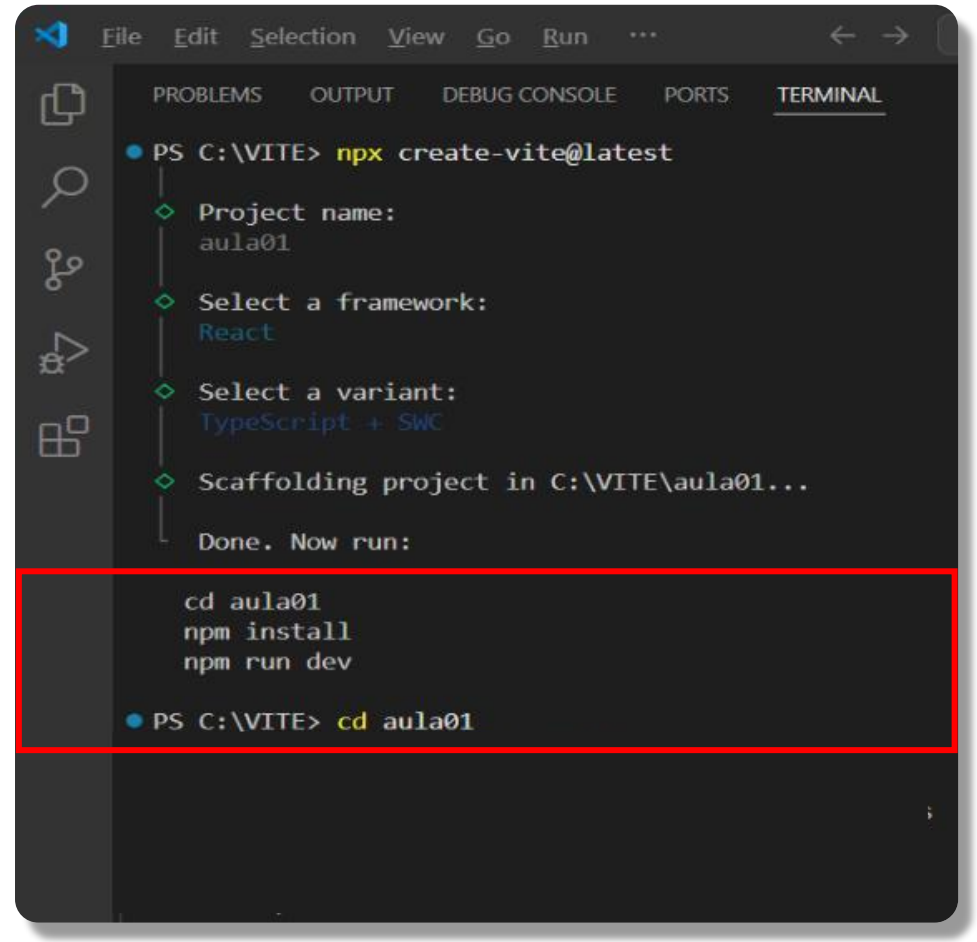
[1] **Transpilar** é o processo de converter código de uma linguagem de programação para outra linguagem de alto nível, mantendo a mesma funcionalidade. É como uma "tradução" entre linguagens que os humanos entendem bem — diferente de compilar, que transforma o código em linguagem de máquina.

VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
3. Vamos selecionar o *framework* que iremos criar o nosso projeto.
4. Agora vamos selecionar a 'linguagem' que iremos trabalhar.
5. O *scaffolding* do nosso projeto foi criado, vamos agora, seguir os três passos descritos no retorno da criação, o primeiro passo é acessar a pasta do nosso projeto, no prompt do Terminal digitamos o nome do projeto que acabamos de criar:

   - ✓ `cd aula01`

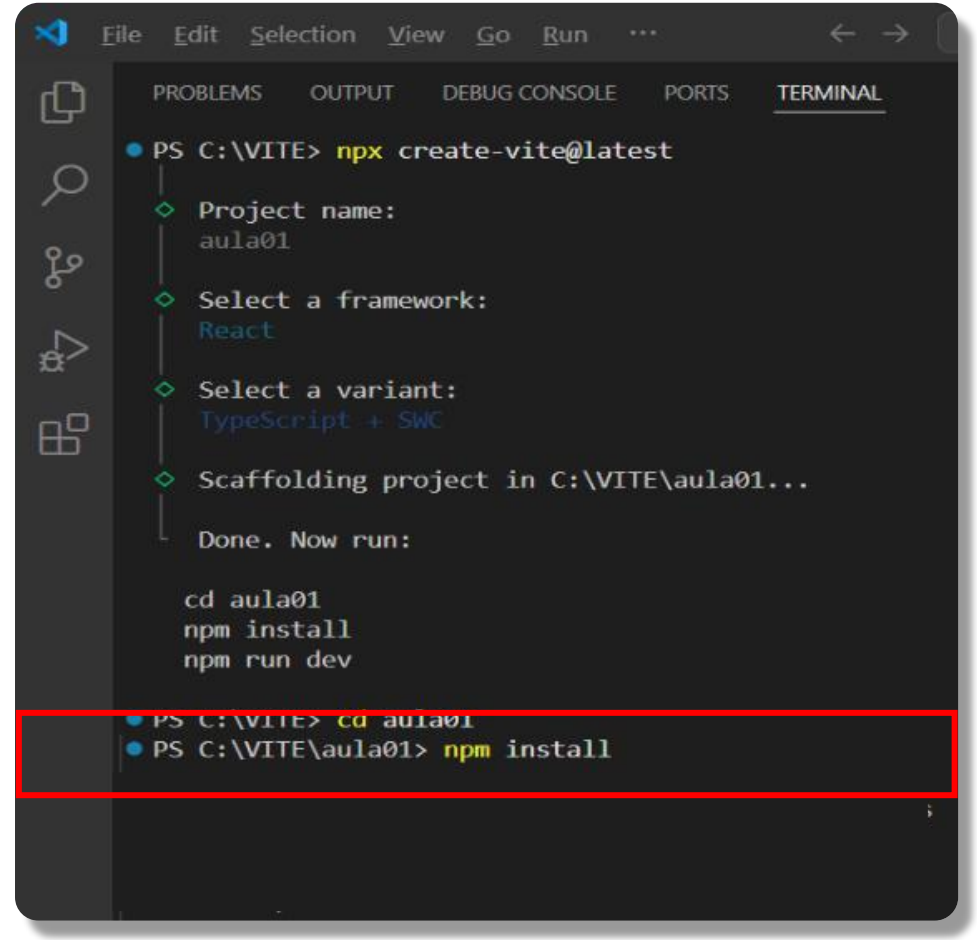


VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
3. Vamos selecionar o *framework* que iremos criar o nosso projeto.
4. Agora vamos selecionar a 'linguagem' que iremos trabalhar.
5. O scaffolding do nosso projeto foi criado, vamos digitar no prompt do Terminal o nome do projeto que criamos:
6. Na sequência digitamos a instrução 'npm install' para que os pacotes do react + vite + typescript + swc sejam de fato instalados em nossa aplicação. Para tanto, digitamos no prompt do Terminal o nome a seguinte instrução:

- ✓ `npm install`

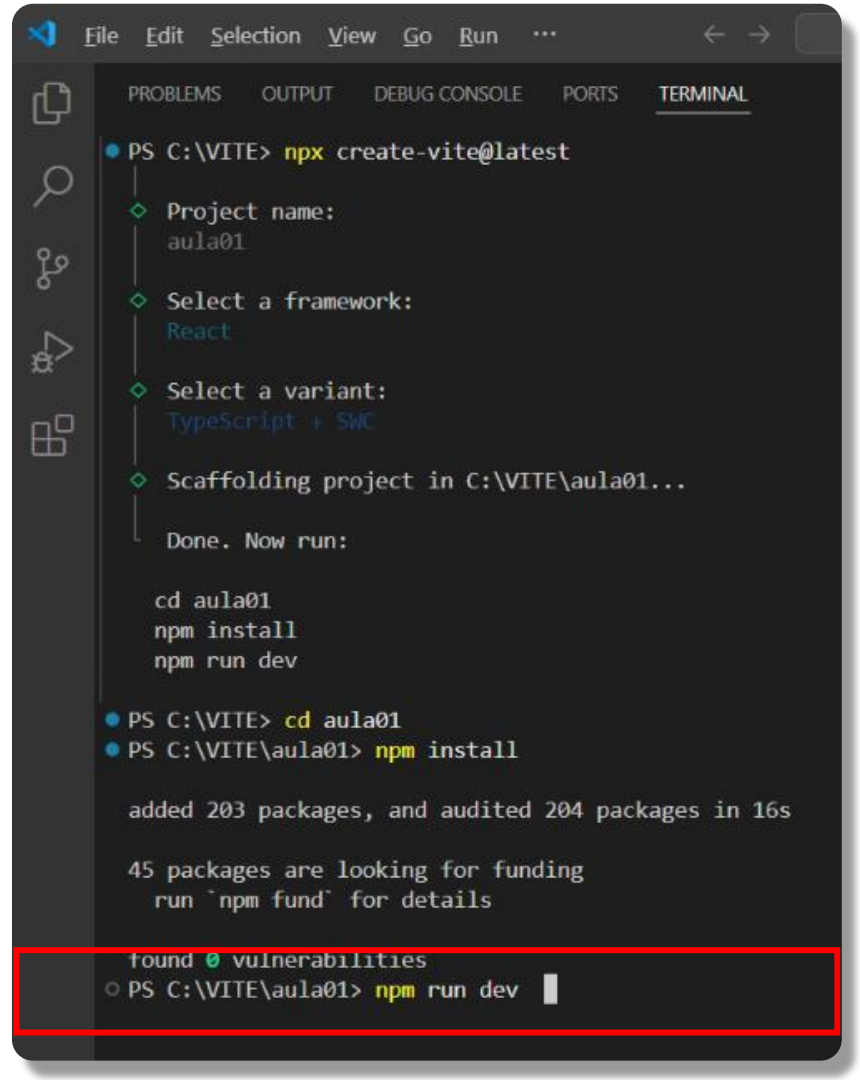


VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:
2. Vamos informar o nome do projeto a ser criado.
3. Vamos selecionar o *framework* que iremos criar o nosso projeto.
4. Agora vamos selecionar a 'linguagem' que iremos trabalhar.
5. O scaffolding do nosso projeto foi criado, vamos digitar no prompt do Terminal o nome do projeto que criamos:
6. Na sequência digitamos a instrução 'npm install' para que os pacotes do react + vite + typescript + swc sejam de fato instalados em nossa aplicação. Para tanto, digitamos no prompt do Terminal o nome a seguinte instrução:
7. No último passo, devemos digitar 'npm run dev' para executar nossa aplicação

- ✓ `npm run dev`



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

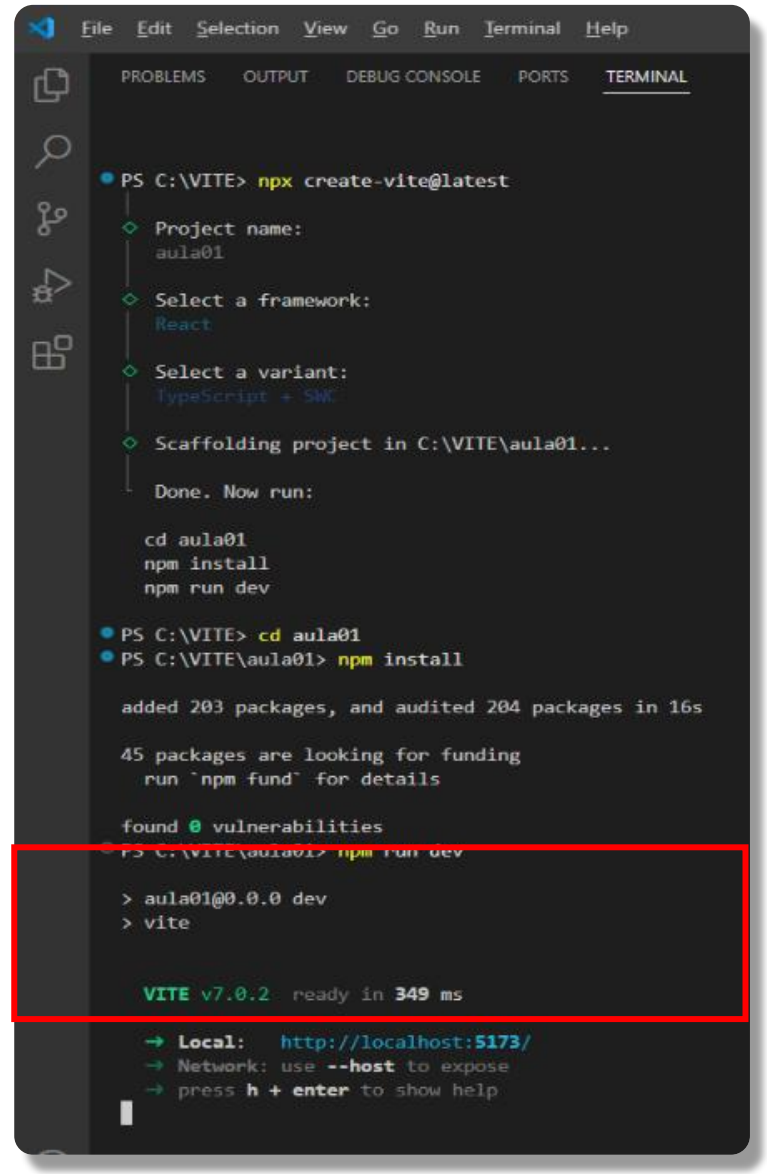
- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:

...;

7. No último passo, devemos digitar 'npm run dev' para executar nossa aplicação.

8. A resposta recebida corresponde ao seguinte:
   - ✓ > aula01@0.0.0 dev
     - ✓ aula01: é o nome do seu projeto
     - ✓ 0.0.0: é a versão atual do seu projeto
     - ✓ dev: é o script que você está executando (developer)



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

1. No terminal aberto vamos digitar o seguinte:

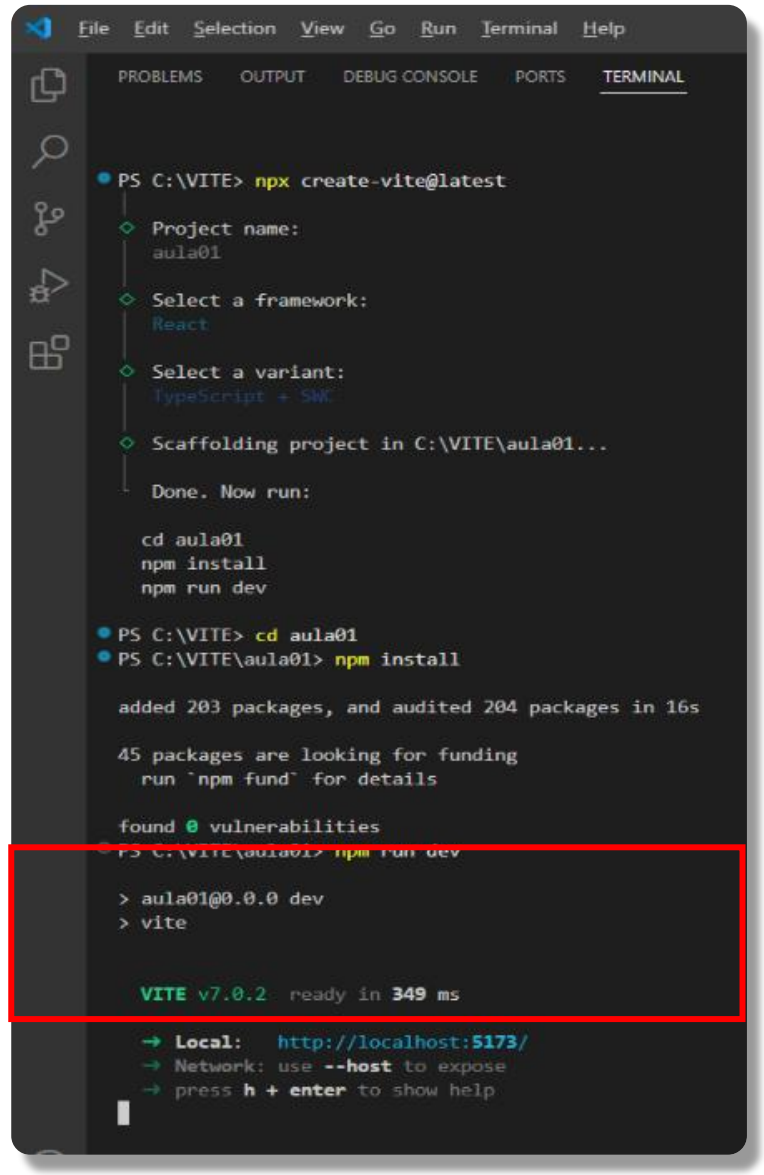
...;

7. No último passo, devemos digitar 'npm run dev' para executar nossa aplicação.

8. A resposta recebida corresponde ao seguinte: > aula01@0.0.0 dev

   - ✓ > vite

     - ✓ A linha > vite mostra o comando do script dev que estamos rodando.



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

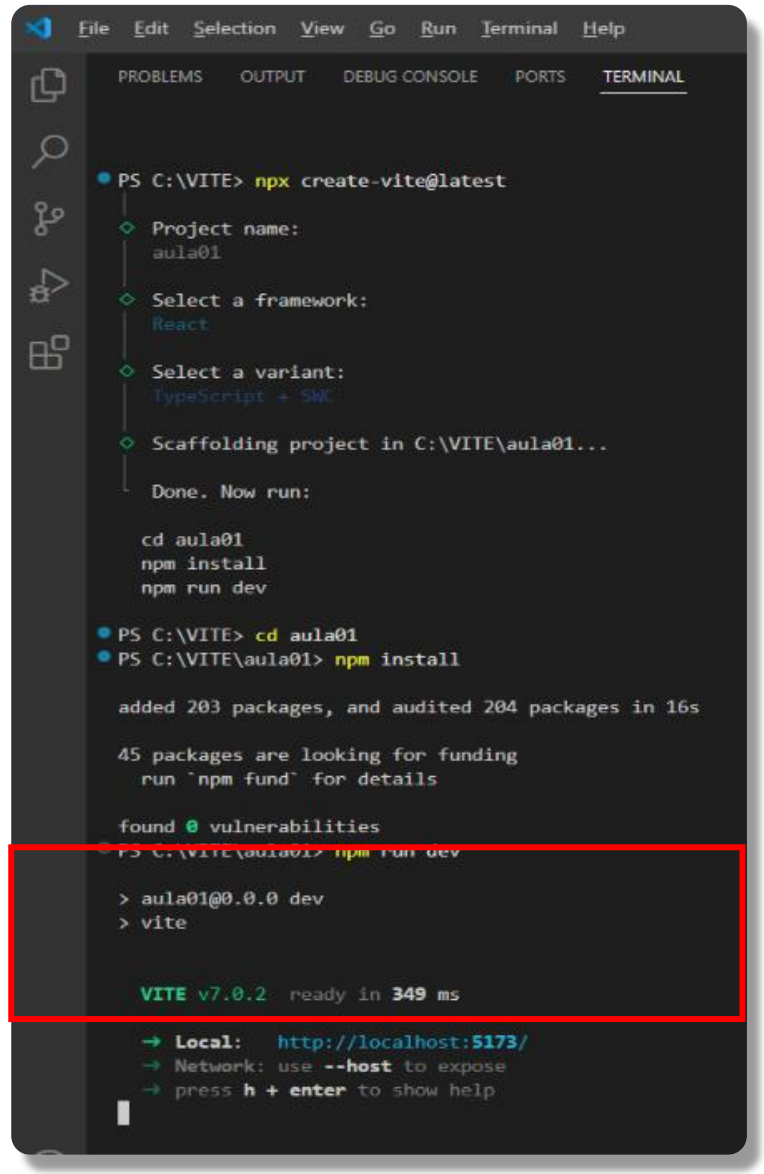
1. No terminal aberto vamos digitar o seguinte:

...;

7. No último passo, devemos digitar 'npm run dev' para executar nossa aplicação.

8. A resposta recebida corresponde ao seguinte:

   - ✓ VITE v7.0.2 ready in 349 ms
     - ✓ VITE v7.0.2: essa é a versão do Vite que está sendo usada
     - ✓ ready in 349 ms: o Vite ficou pronto em 349 milissegundos (menos de meio segundo!)



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**

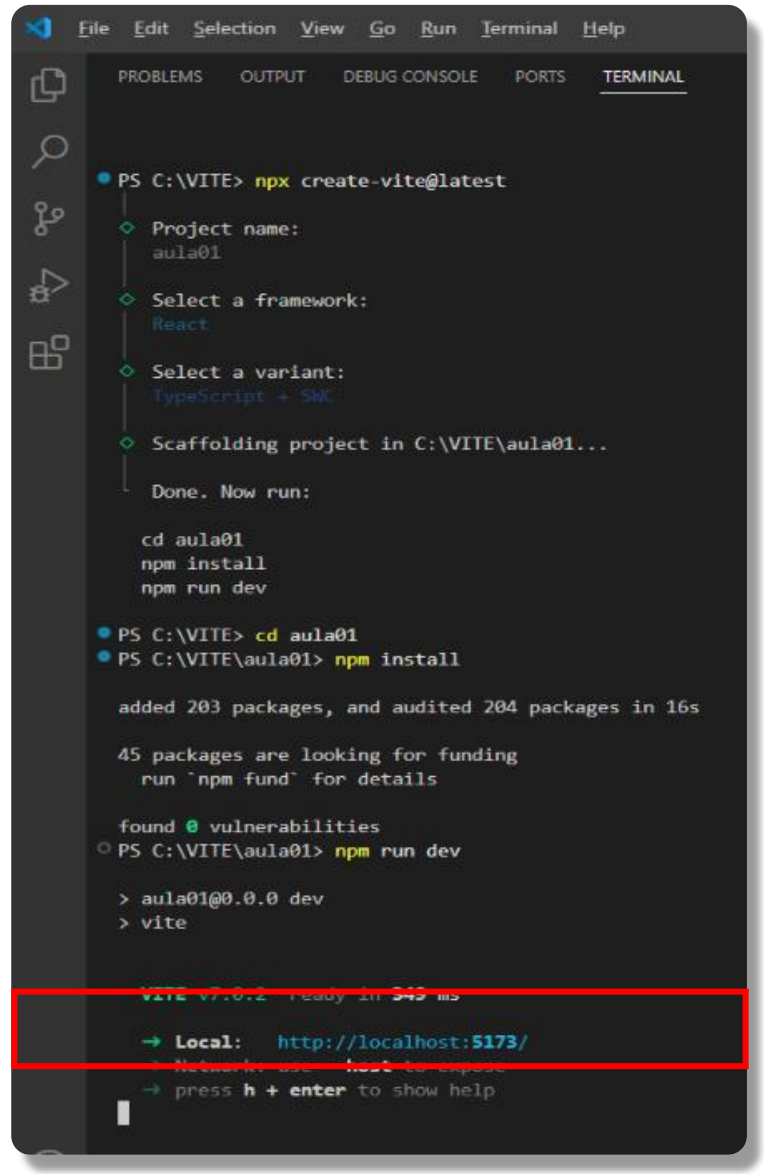
1. No terminal aberto vamos digitar o seguinte:

...;

7. No último passo, devemos digitar 'npm run dev' para executar nossa aplicação.

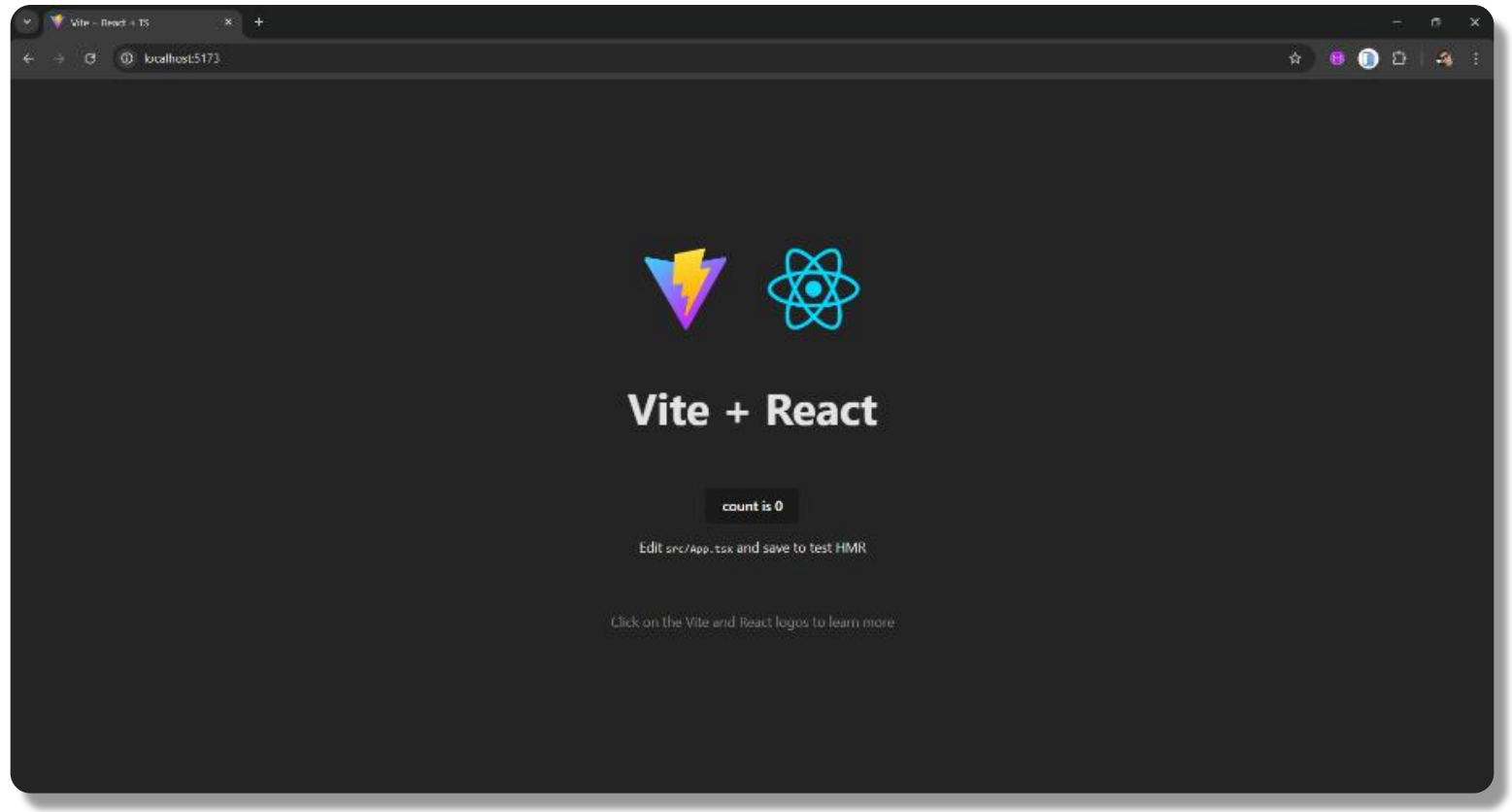
8. A resposta recebida corresponde ao seguinte:

- ✓ ➜ Local: http://localhost:5173/
  - ✓ Isso é o link onde seu site está rodando localmente.
  - ✓ localhost: significa que está só no seu computador
  - ✓ 5173: é a porta onde o Vite está servindo sua aplicação



VITE – DA CRIAÇÃO DO PROJETO AO PRIMEIRO RUN LOCAL

- ✓ **Passos iniciais**
  - ✓ Acessando o endereço http://localhost:5173/ no navegador, devemos ter a seguinte resposta



VITE – EXPLORANDO A ESTRUTURA DO PROJETO

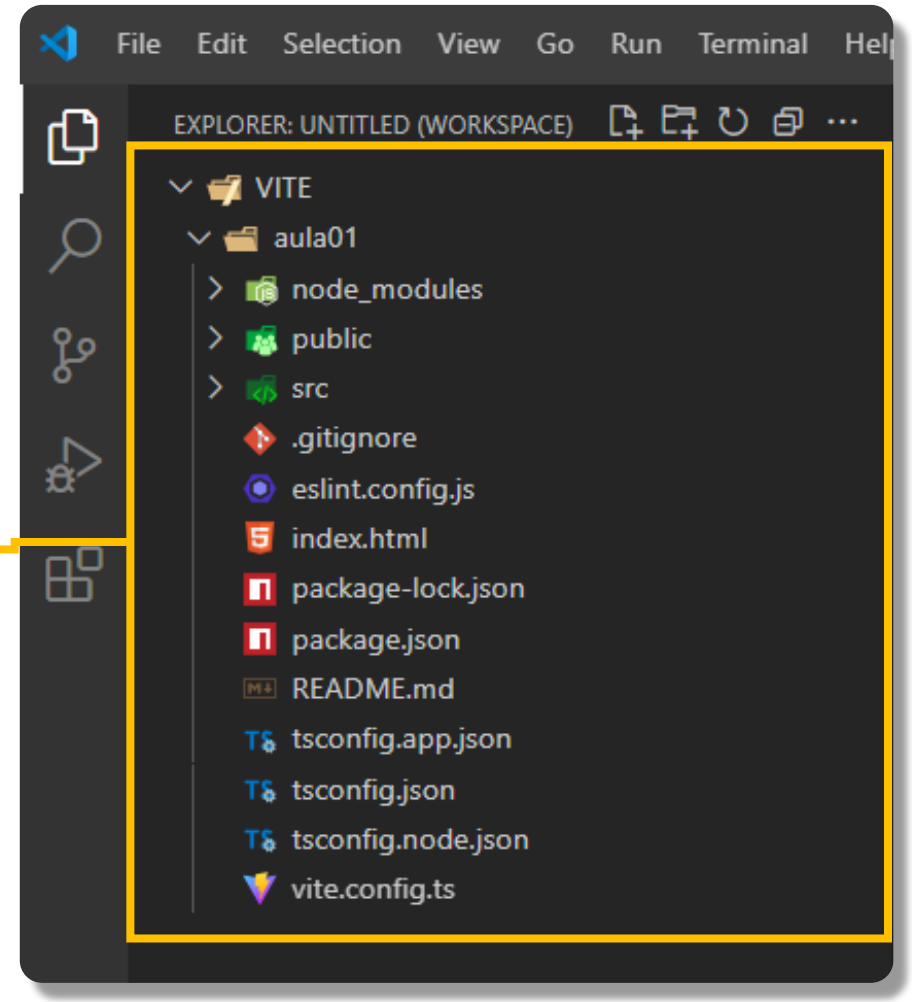
VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

VITE/AULA01

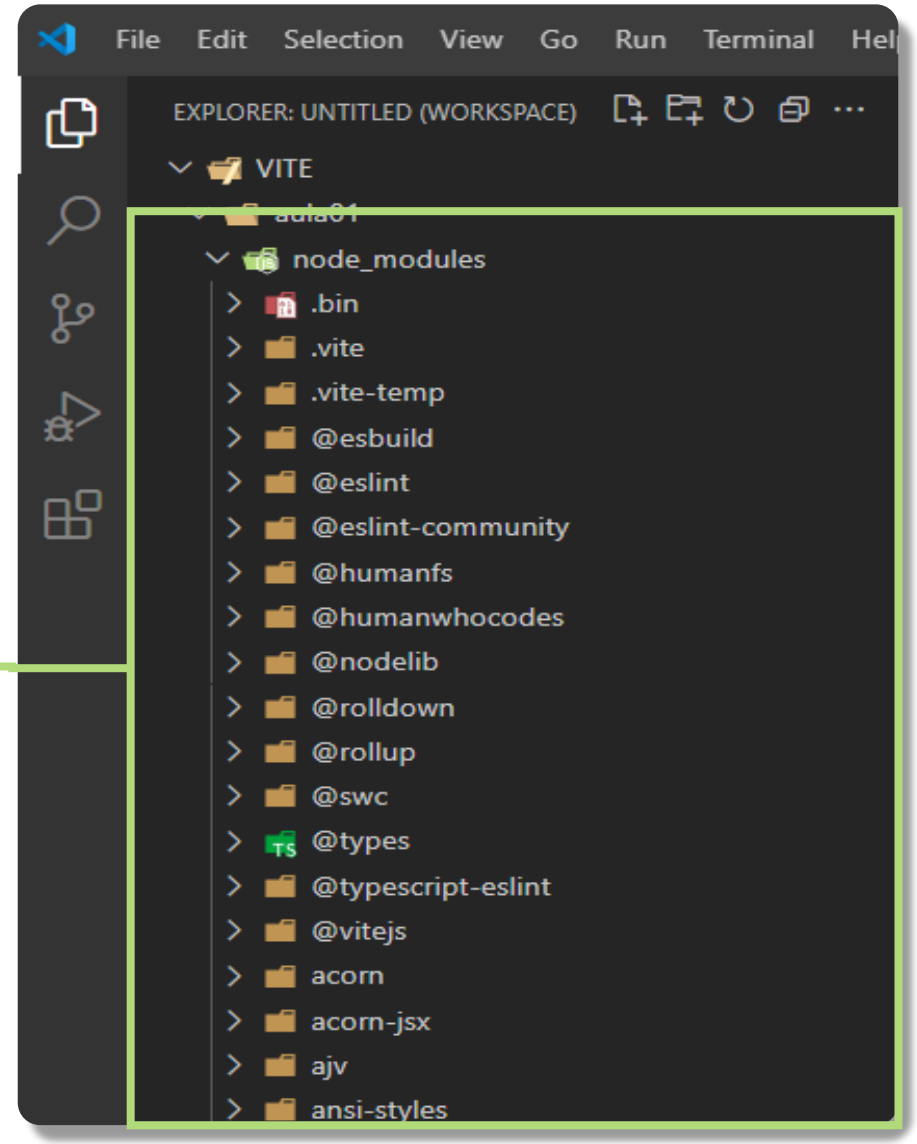
Durante a criação do projeto, foram gerados automaticamente mais de **3.700** arquivos, somando cerca de **120 MB**.

Essa quantidade se deve, principalmente, à instalação das dependências do Vite e do React, que compõem a base para o funcionamento da aplicação e facilitam o desenvolvimento.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



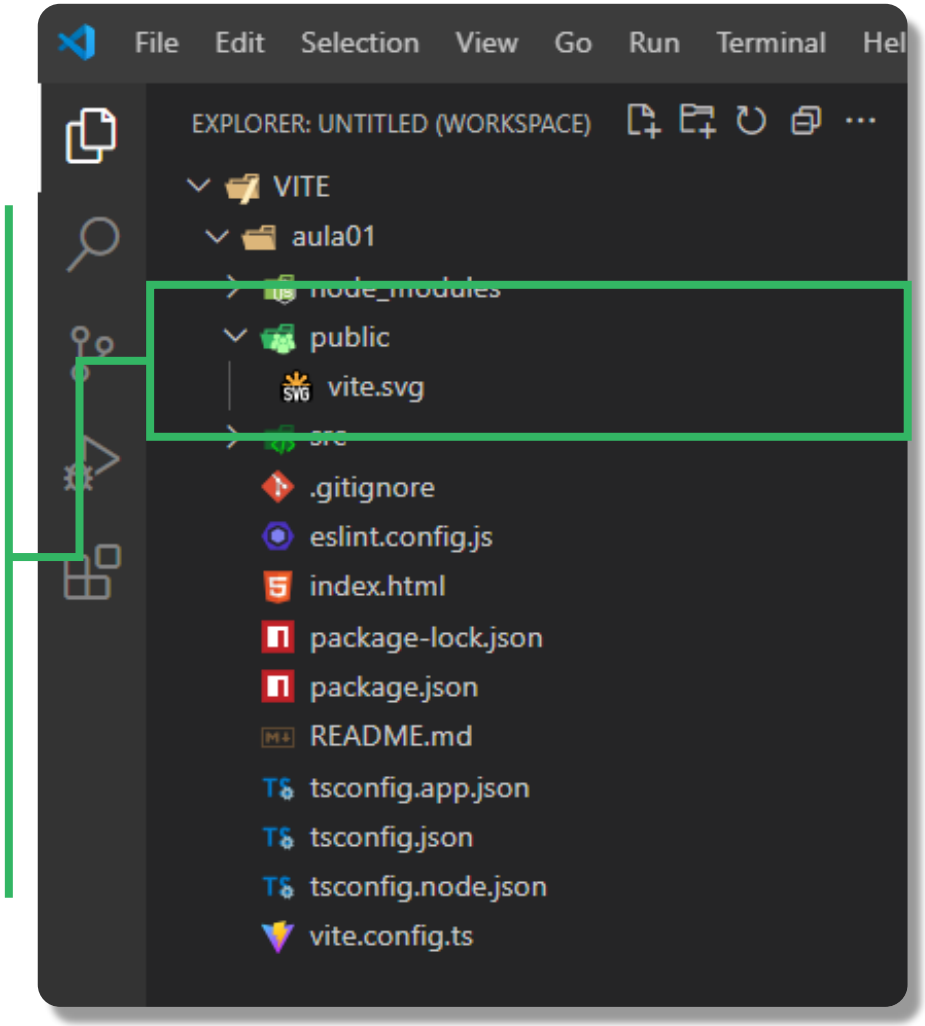
node_modules

São bibliotecas do Node que estão disponível para uso na aplicação.

Essa pasta **NÃO** deverá ser copiada no transporte da sua aplicação entre computadores diferentes.

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



public

Contém os arquivos estáticos que serão carregados no lado do cliente.

Diretório para arquivos estáticos públicos, como imagens, fontes, favicons, etc.

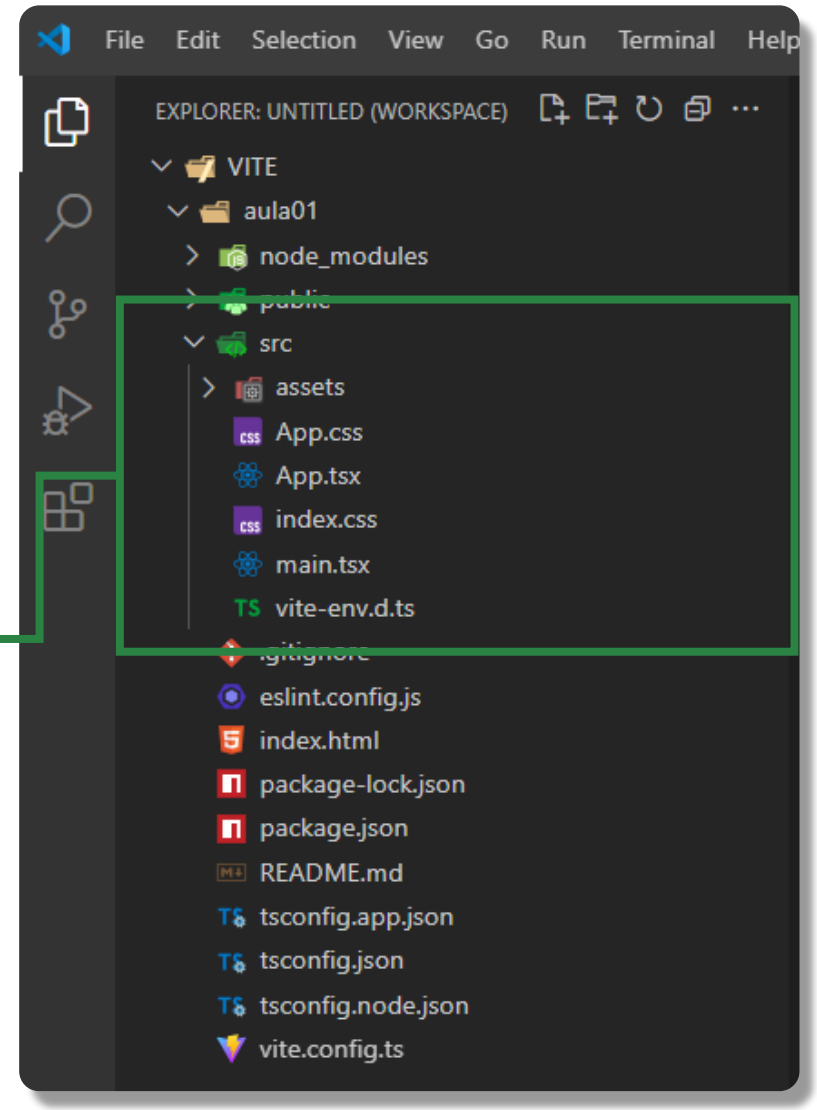
Os arquivos aqui são servidos diretamente no navegador.

Exemplo: Se houver um arquivo em public/logo.png, você pode acessá-lo via /logo.png.

Não é necessário usar import para acessar arquivos dessa pasta.

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



src

Nesta pasta serão armazenado arquivos que serão carregados do lado do servidor.

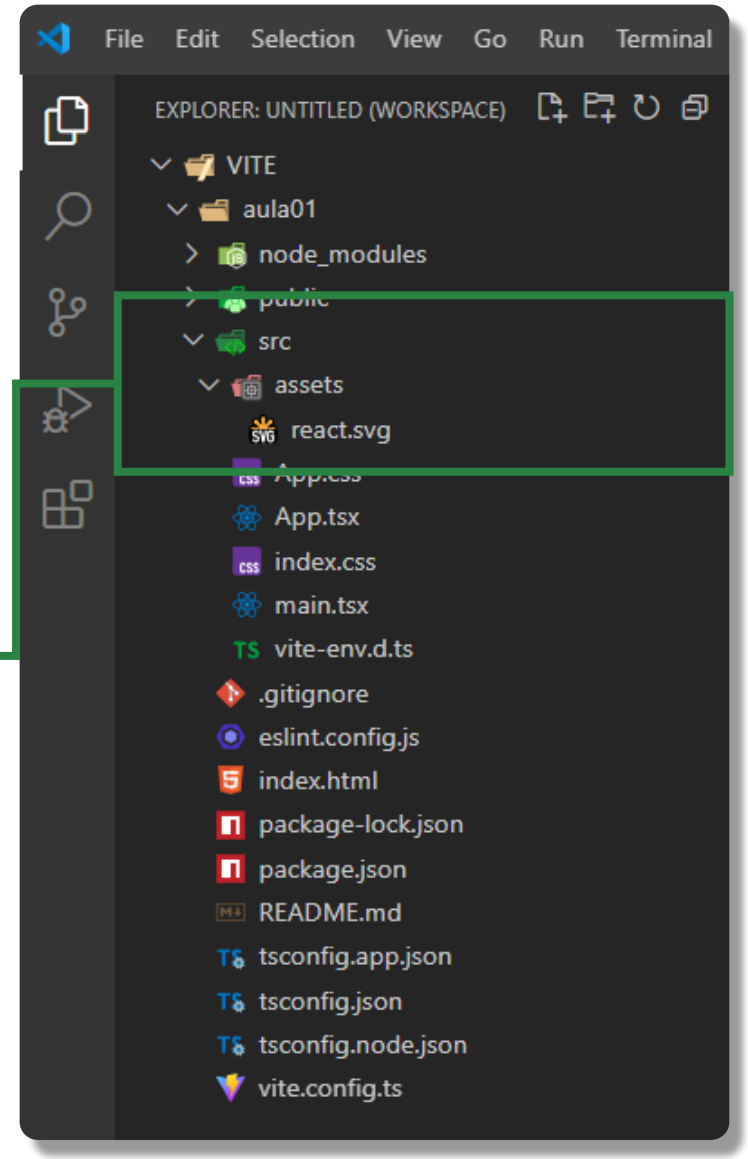
Um padrão recomendado para projetos maiores, mas opcional.

Quando usado, a pasta src/ geralmente contém todas as pastas como pages/ (ou app/), components/, styles/, e outras.

Ajuda a diferenciar o código-fonte da configuração do projeto.

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



src/assests

É um diretório convencional usado em projetos *frontend* (como React, Vite, Vue, Angular) para armazenar recursos estáticos utilizados pela aplicação.

O que normalmente vai em **src/assets**?

- Imagens (.jpg, .png, .svg, .webp)
- Ícones (como arquivos .svg ou .ico)
- Fontes personalizadas (.ttf, .woff, .woff2)
- Vídeos ou áudios utilizados na interface

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

src/App.css

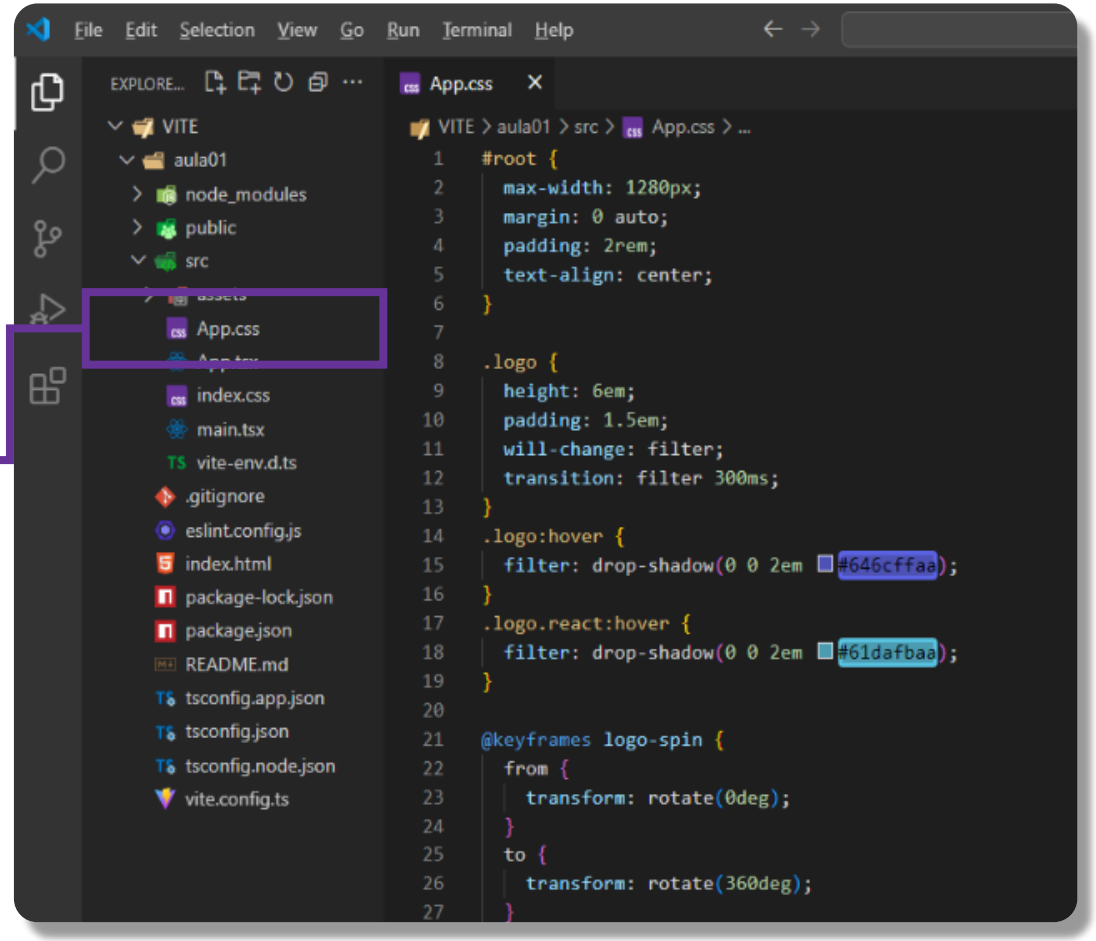
É o arquivo de estilos principal ligado ao componente raiz (App.jsx ou App.tsx) em projetos React.

Ele define estilos globais e visuais iniciais da interface.

Ao ser importado, aplica os estilos globalmente, já que CSS tradicional não possui escopo local.

Para estilos por componente, usa-se CSS Modules ou Tailwind.

Boas práticas incluem padronizar estilos globais, evitar excessos no App.css e organizar estilos em arquivos específicos para projetos maiores.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

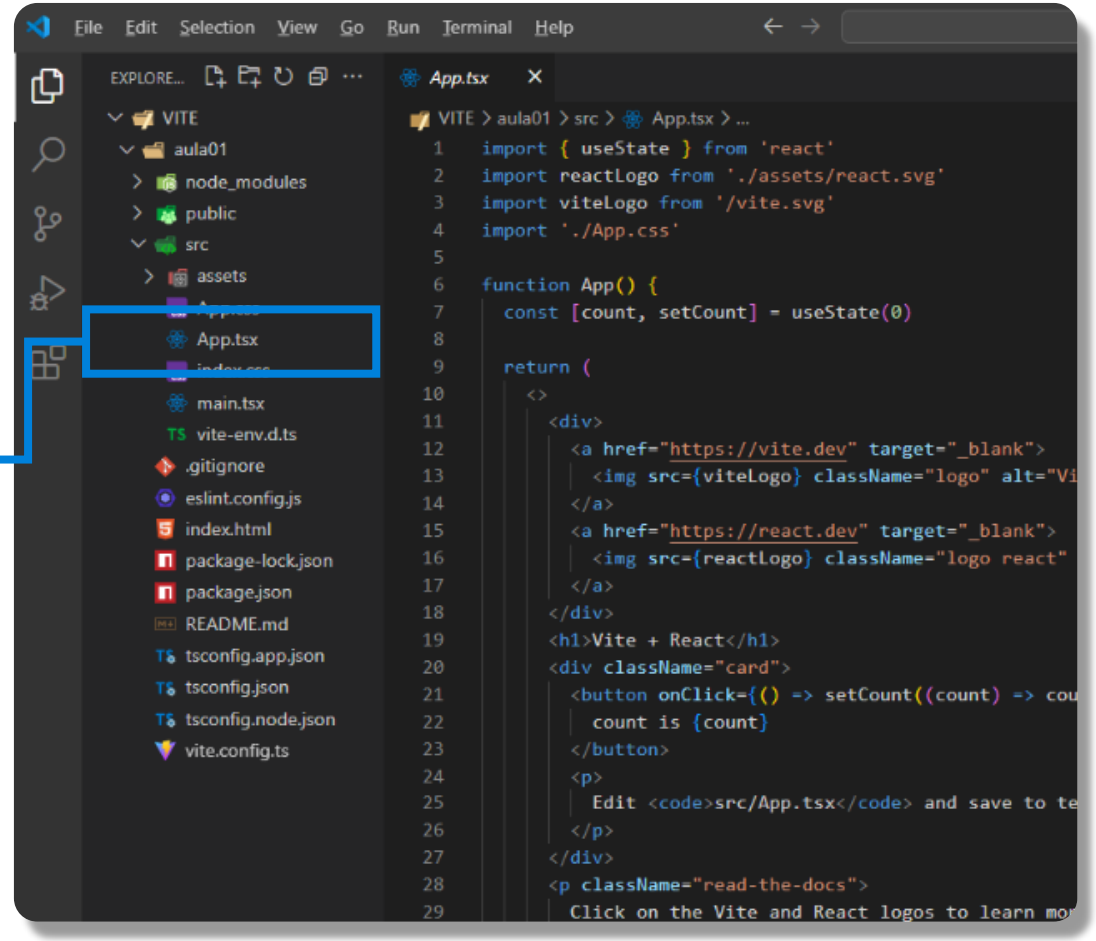
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

src/App.tsx

O App.tsx é o componente principal de uma aplicação React com TypeScript e atua como o ponto de entrada da interface.

Nele são definidos a estrutura base da aplicação, a renderização de componentes filhos (como cabeçalho, menus e páginas), a importação de estilos globais (como o App.css) e a organização da navegação e dos estados globais.

Justamente por isso, é o local ideal para incluir componentes que se repetem em todas as páginas, como Header, Footer e Menu, garantindo consistência visual em toda a aplicação.



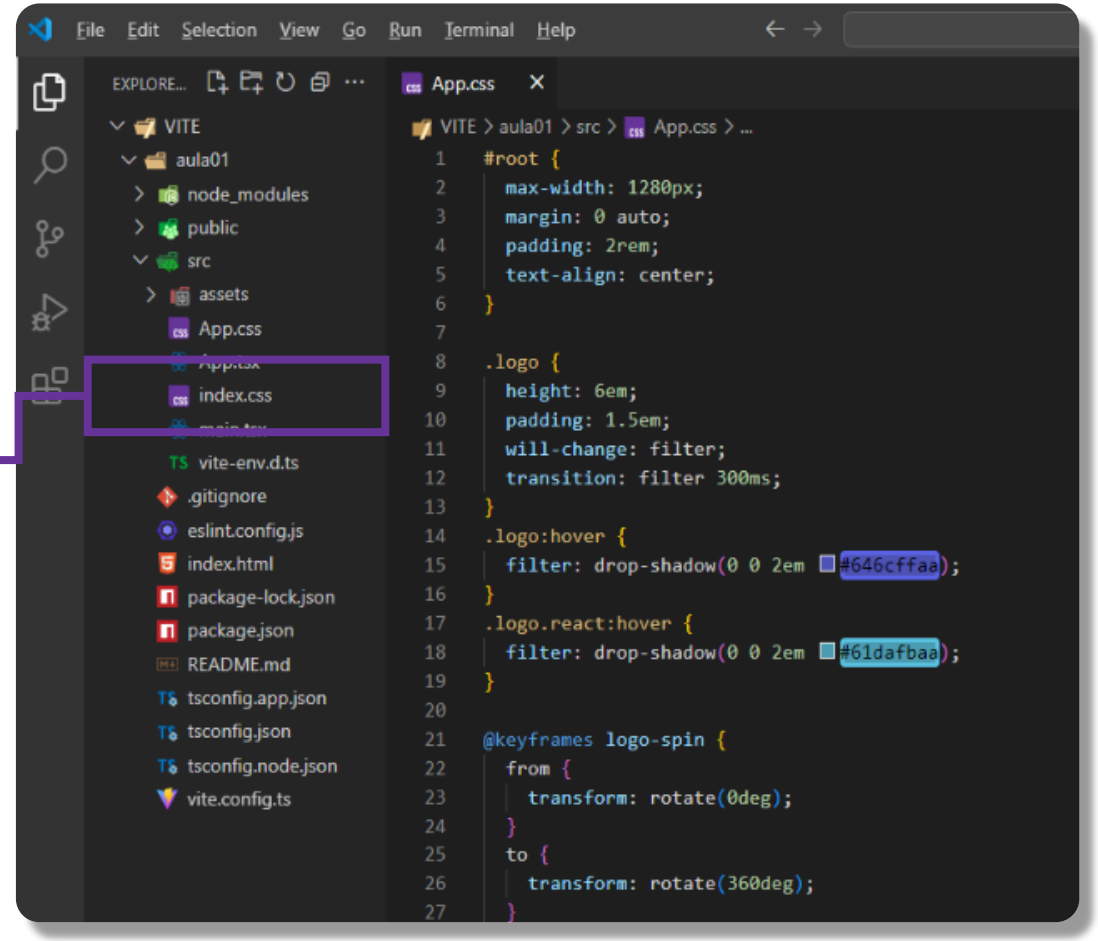
VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

src/index.css

Define os estilos globais da aplicação React. Ele é normalmente importado no main.tsx ou main.jsx e serve para aplicar regras gerais como resets de CSS, fontes padrão, cores de fundo e estilos para html e body. Assim, garante uma base visual consistente em toda a aplicação.

Enquanto o App.css estiliza apenas o componente principal, o index.css afeta a aplicação inteira. É ideal para configurações amplas e reutilizáveis. Em projetos maiores, costuma-se manter esse arquivo enxuto, concentrando estilos específicos em arquivos separados por componente para facilitar a manutenção.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

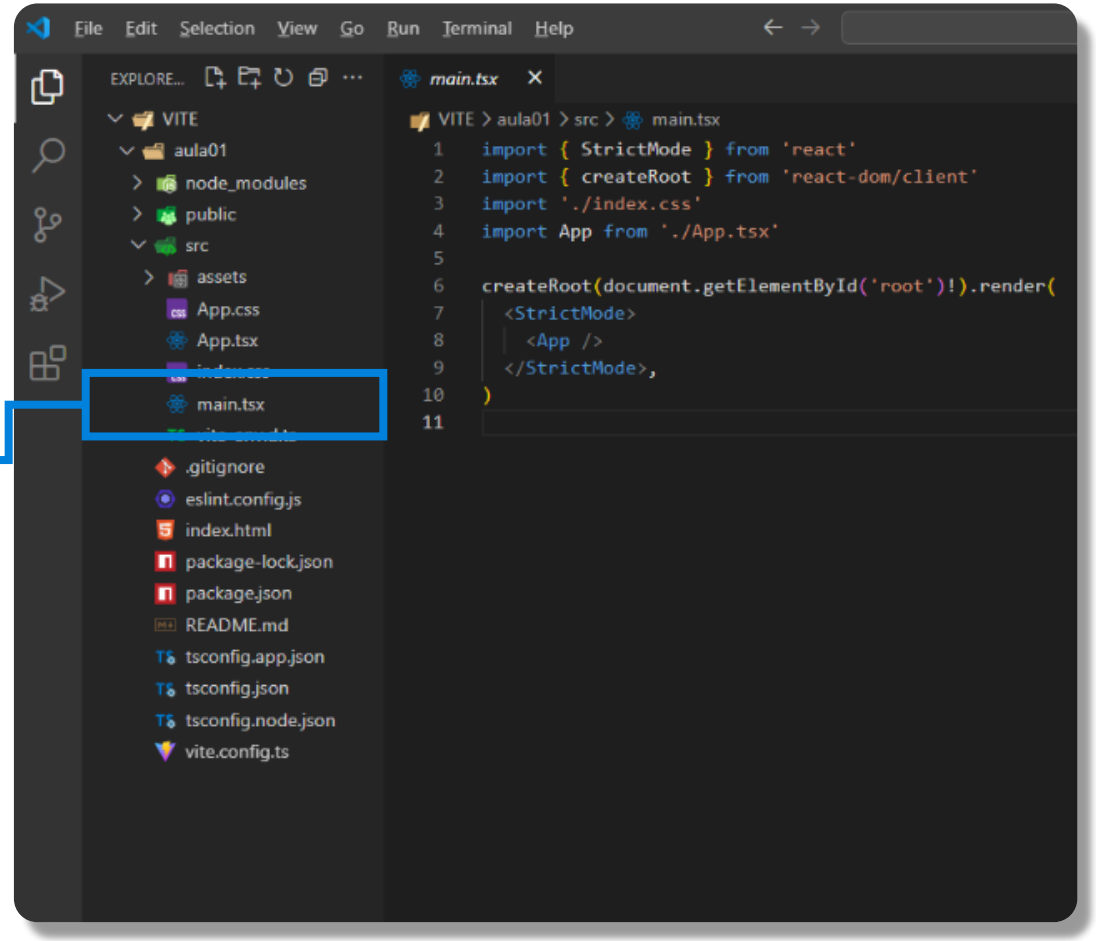
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

src/main.tsx

É o ponto de entrada de uma aplicação React com TypeScript, responsável por inicializar a interface e conectá-la ao DOM.

Nesse arquivo, o React importa o componente principal (App.tsx) e o renderiza dentro da div#root do index.html usando ReactDOM.createRoot(...).

É comum também importar estilos globais, como o index.css, e configurar bibliotecas de contexto, roteadores ou temas globais.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

Exemplo de Quadro Comparativo entre os arquivos main.tsx e App.tsx

Característica	main.tsx	App.tsx
Função principal	Inicializa a aplicação e conecta ao DOM	Define a estrutura visual e lógica da interface
Papel no projeto	Ponto de entrada da aplicação	Componente raiz da interface
Local onde atua	Monta o React dentro da div#root do index.html	Renderiza os componentes como Header, Footer, Rotas etc.
Importa	App.tsx, index.css, bibliotecas globais	Componentes visuais e páginas da aplicação
Tipagem e extensão	Usa .tsx com JSX + TypeScript	Também usa .tsx, com JSX tipado
Frequência de edição	Editado apenas em configurações iniciais	Editado com frequência durante o desenvolvimento
Relação entre eles	Chama o App.tsx	É chamado por main.tsx

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

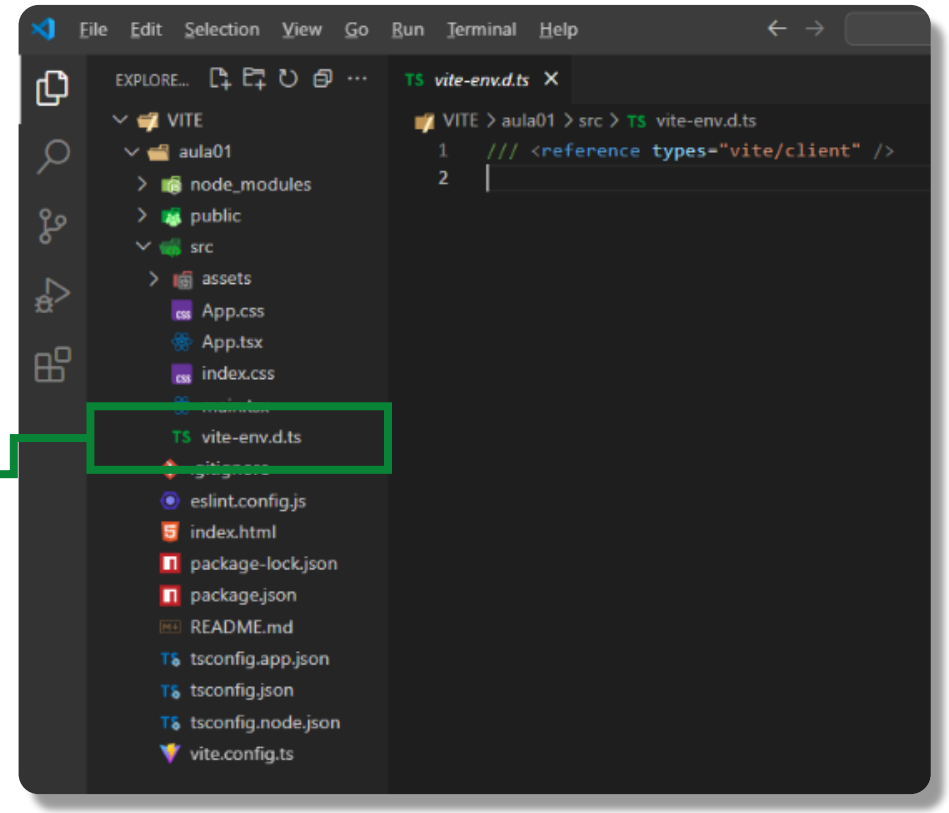
src/vite-env.ts

É uma declaração de tipos usada em projetos TypeScript com Vite.

Permite que o TypeScript reconheça corretamente variáveis e recursos específicos do ambiente Vite, como import.meta.env, onde ficam armazenadas as variáveis de ambiente.

Isso garante que o editor e o compilador tenham acesso às tipagens corretas durante o desenvolvimento. Além disso, o arquivo importa definições da biblioteca vite/client, permitindo a tipagem adequada para recursos estáticos como imagens, arquivos .svg e estilos.

Criado automaticamente ao iniciar um projeto com Vite, esse arquivo geralmente não precisa ser modificado, mas é fundamental para o funcionamento correto da tipagem no ambiente de desenvolvimento.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

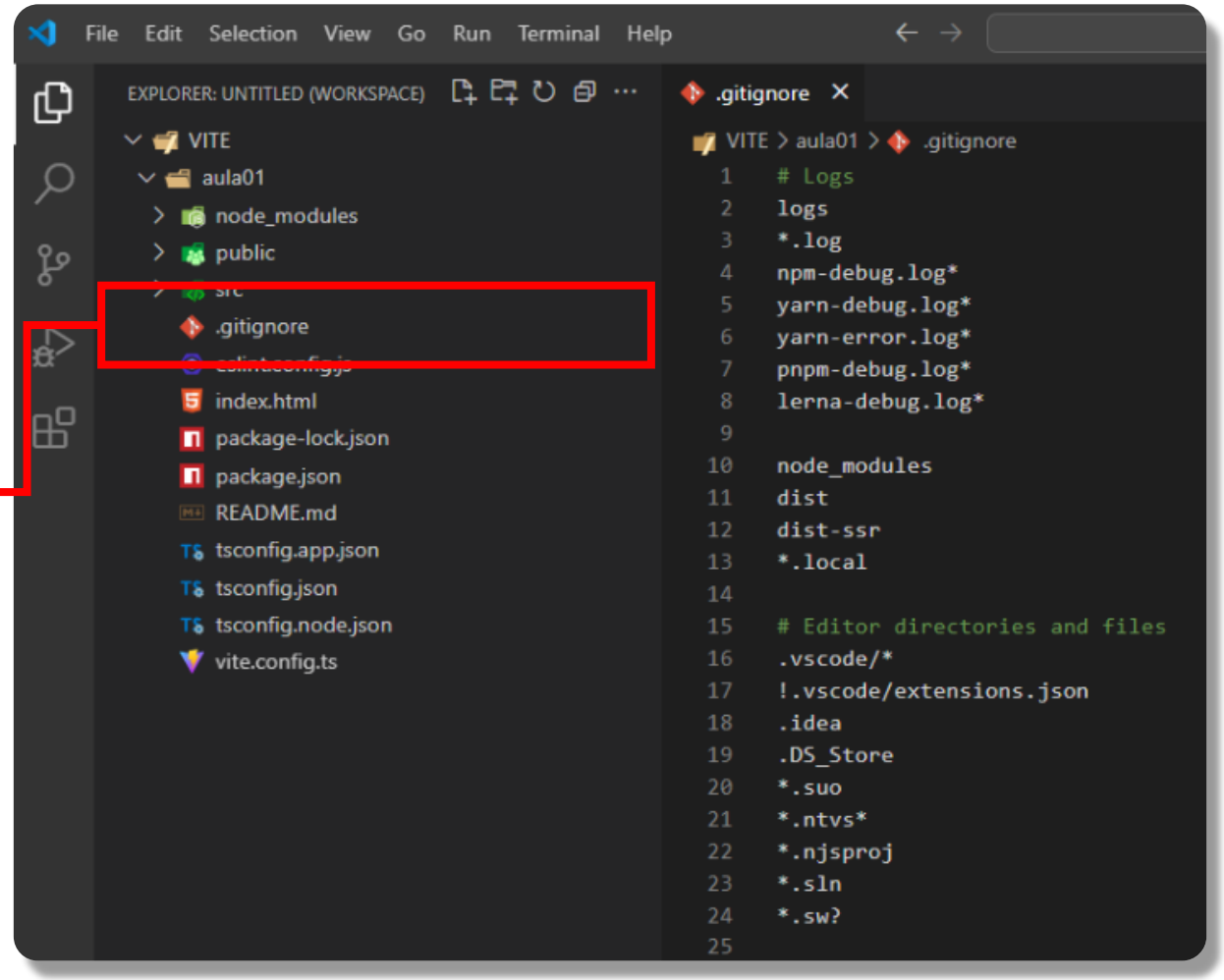
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

.gitignore

Este arquivo serve para informar ao GIT quais arquivos e pastas devem ser ignorados no versionamento.

Isso inclui arquivos temporários, pastas de dependências como node_modules, configurações locais, arquivos de build ou qualquer outro conteúdo que não deve ser enviado para o repositório.

Dessa forma, o .gitignore ajuda a manter o repositório limpo, leve e focado apenas no que é realmente necessário para o projeto.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

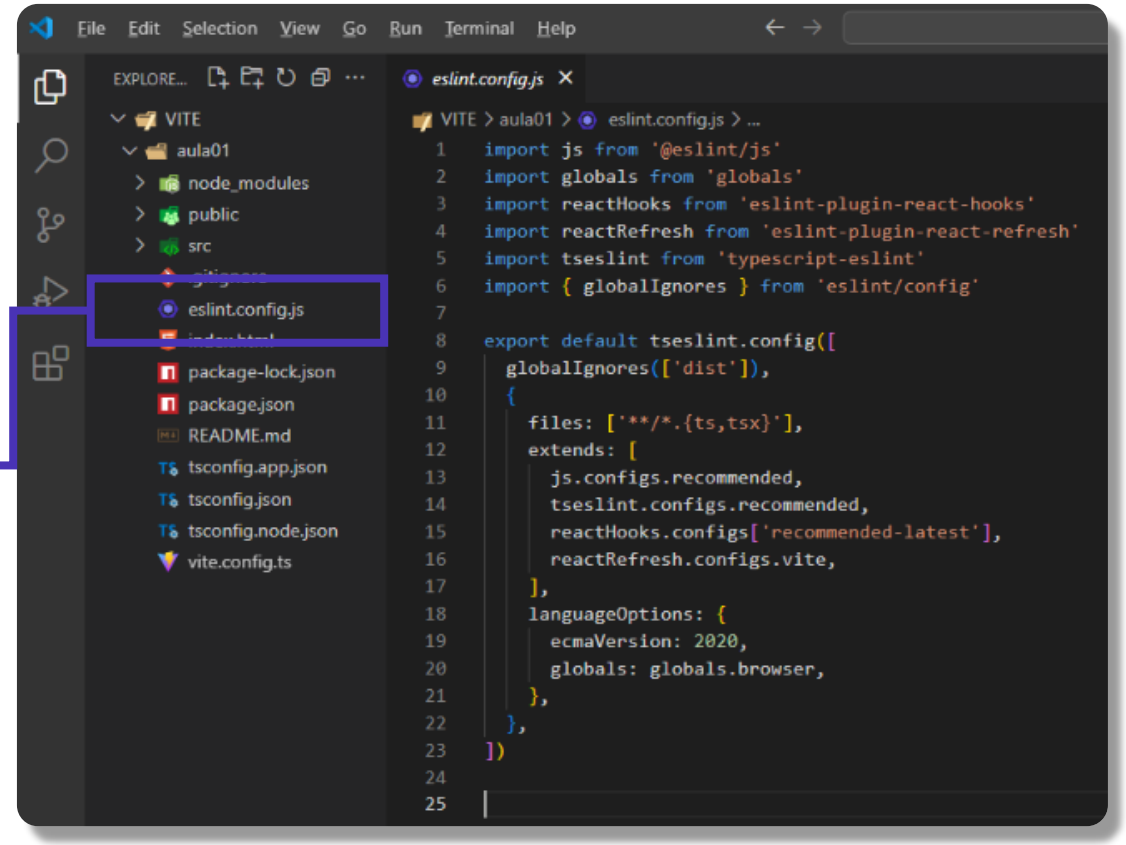
eslint.config.js

O arquivo é usado para configurar o ESLint, uma ferramenta que analisa o código JavaScript ou TypeScript em busca de erros, problemas de estilo e más práticas de programação.

Define quais regras o projeto seguirá, como indentação, uso de ponto e vírgula, declarações não utilizadas, entre outras.

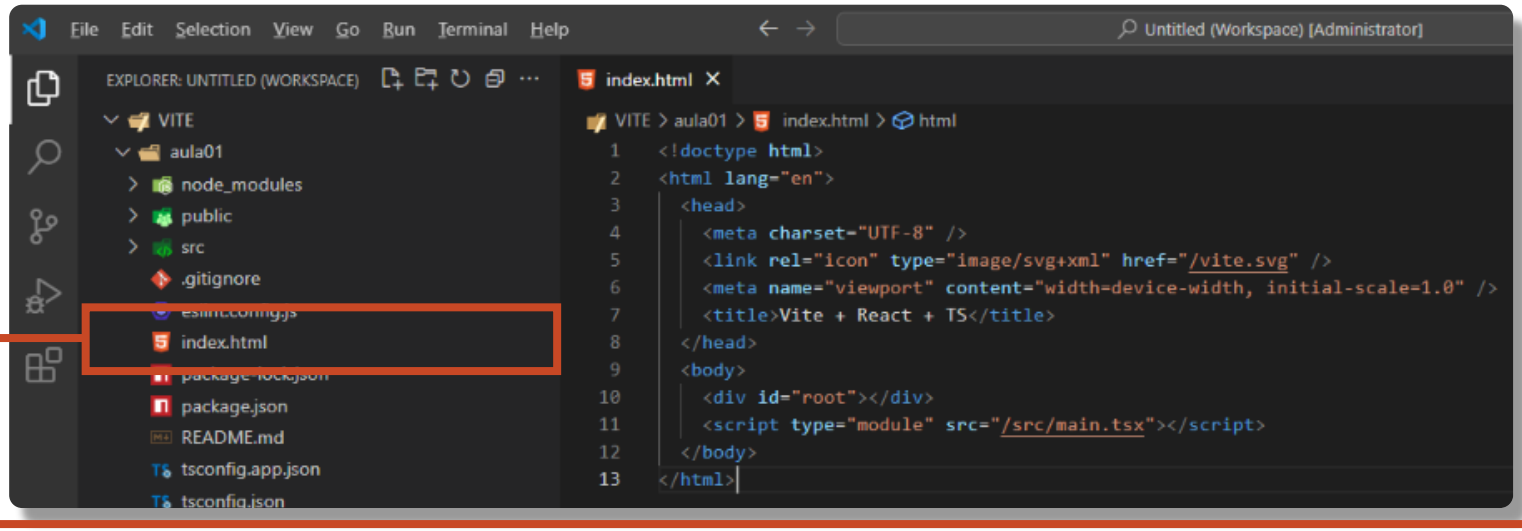
Ele também pode incluir plugins e extensões específicas, como para React ou TypeScript, e determinar quais arquivos ou pastas devem ser verificados.

Ajuda a manter o código limpo, padronizado e mais fácil de manter, especialmente em equipes de desenvolvimento.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



index.html

O arquivo index.html é a porta de entrada da aplicação web. Ele define a estrutura básica da página que será exibida no navegador e carrega o código JavaScript gerado pelo Vite, geralmente apontando para o arquivo **main.js** ou **main.tsx**.

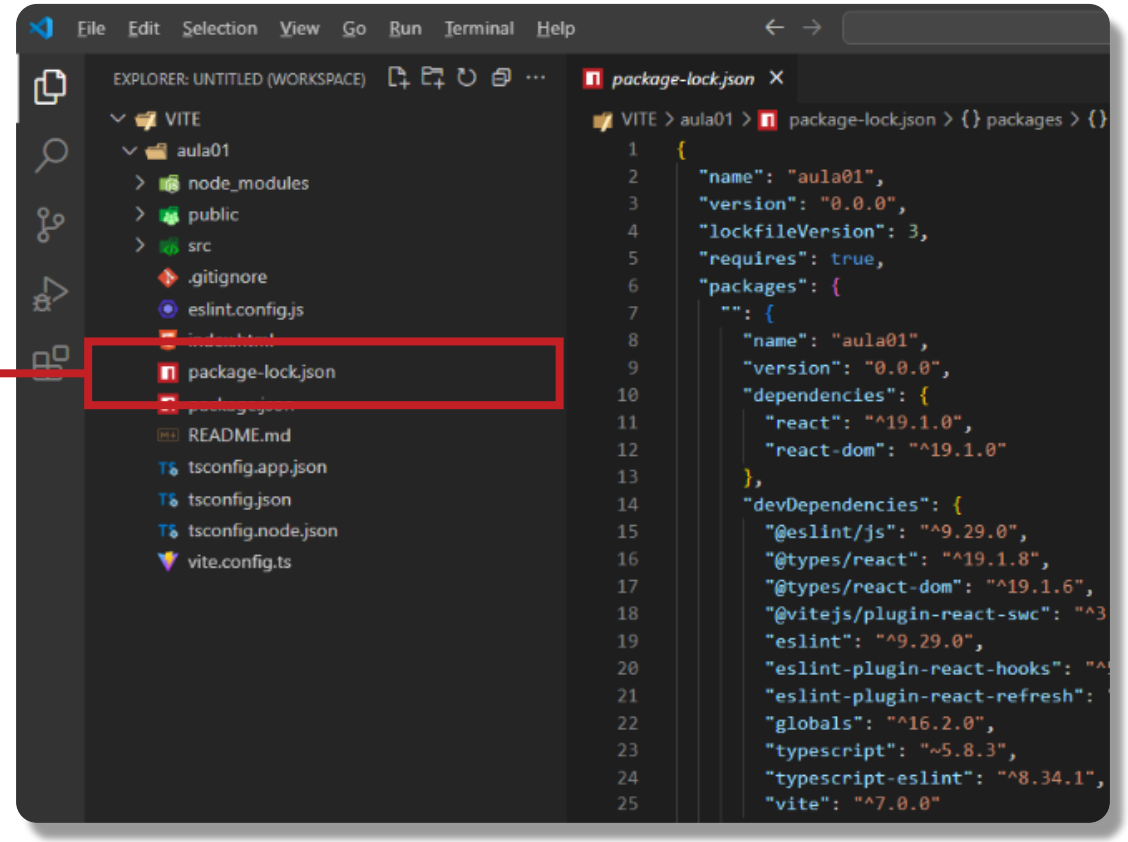
Apesar de o React (ou outro framework) controlar a maior parte da interface via JavaScript, o index.html ainda é essencial para configurar o título da página, links para ícones, fontes, meta tags e o ponto de montagem da aplicação — normalmente um , onde o React renderiza tudo dinamicamente.

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

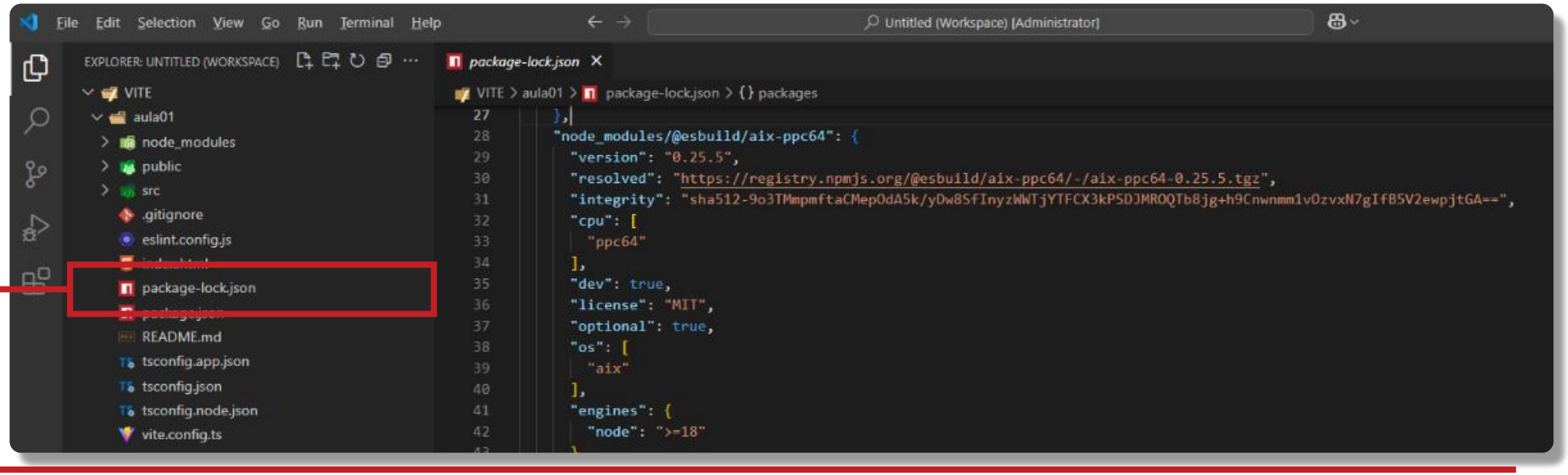
package-lock.json

Este arquivo é gerado automaticamente pelo npm ao instalar as dependências do projeto e tem a função de registrar as versões exatas de cada pacote (e suas dependências internas), garantindo que todos os desenvolvedores e ambientes utilizem a mesma estrutura, evitando conflitos e comportamentos inconsistentes.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.



package-lock.json

Ainda sobre o arquivo **package-lock.json**, a linha **resolved** aponta para a URL exata de onde o pacote foi baixado, normalmente do registro oficial do npm, enquanto a linha **integrity** contém um hash criptográfico (como SHA-512) que permite ao npm verificar se o pacote não foi alterado ou corrompido, assegurando a integridade das dependências.

https://nodejs.reativa.dev/0020-package-lock-json/index

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

package.json

O arquivo package.json é um dos arquivos mais importantes em projetos Node.js, incluindo aplicações frontend com React, Vite, Next.js, etc. Ele atua como "coração" da aplicação, definindo metadados do projeto, como:

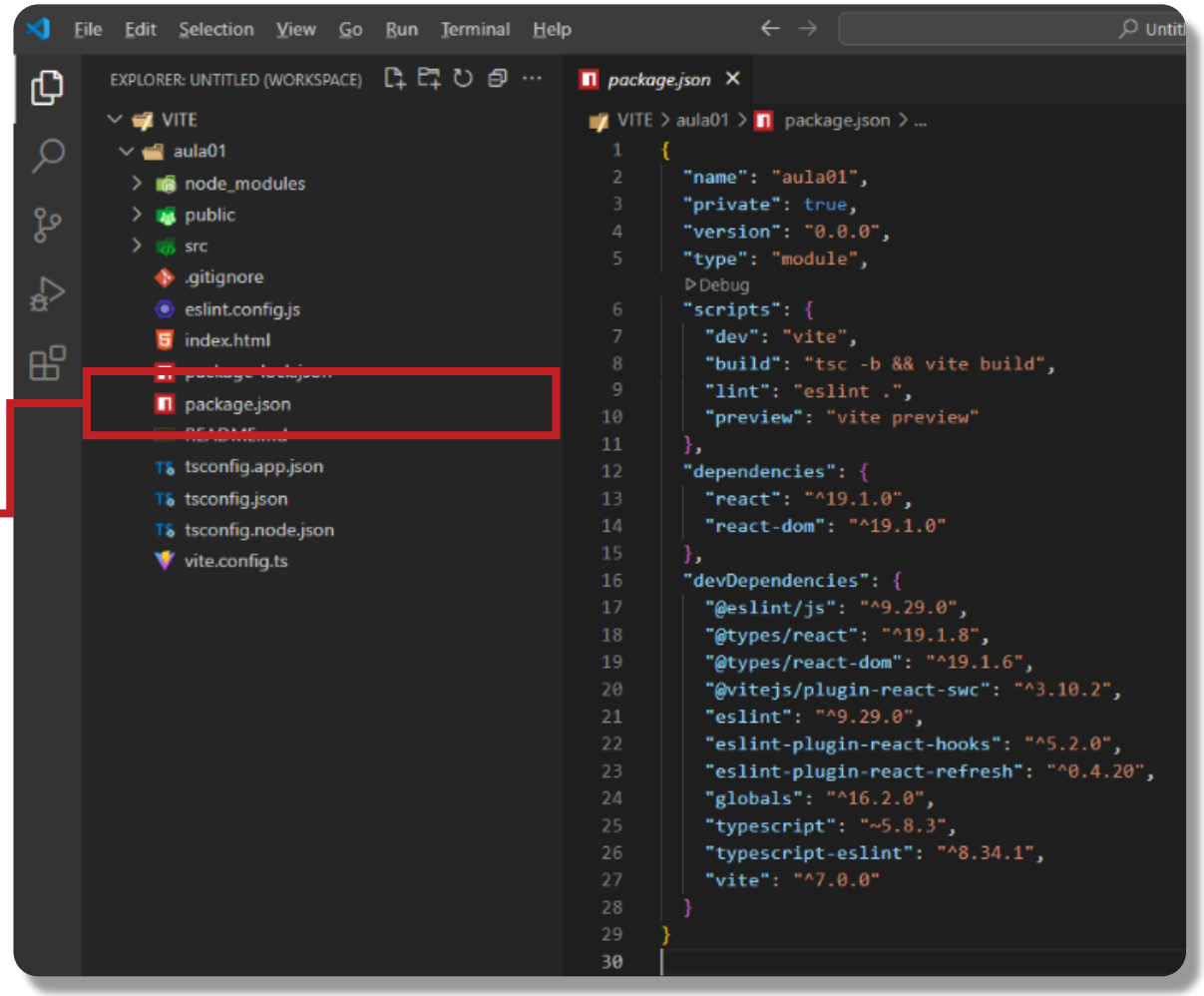
nome e versão do projeto

scripts executáveis (como npm run dev, npm run build)

dependências e dependências de desenvolvimento

licença, autor, e outras informações úteis

https://nodejs.reativa.dev/0019-package-json/index



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

Exemplo de Quadro Comparativo entre os arquivos package.json e package-lock.json

Característica	package.json	package-lock.json
Finalidade	Define as configurações e dependências do projeto	Registra as versões exatas das dependências instaladas
Criação	Criado manualmente ou via npm init	Gerado automaticamente pelo npm após instalação de pacotes
Editável	Sim, é editado diretamente pelo desenvolvedor	Não deve ser editado manualmente
Contém scripts	Sim (start, build, dev, etc.)	Não
Lista de dependências	Sim (com versões aproximadas, ex: ^1.0.0)	Sim (com versões exatas e resolvidas de todas as dependências)
Controle de versão	Pode ser versionado no Git	Deve sempre ser versionado no Git para garantir consistência
Impacto na instalação	Define o que deve ser instalado	Garante que a instalação seja idêntica em diferentes ambientes
Necessário para rodar o projeto	Sim	Sim (para garantir fidelidade do ambiente)

VITE – ANATOMIA DO PROJETO INICIAL COM VITE

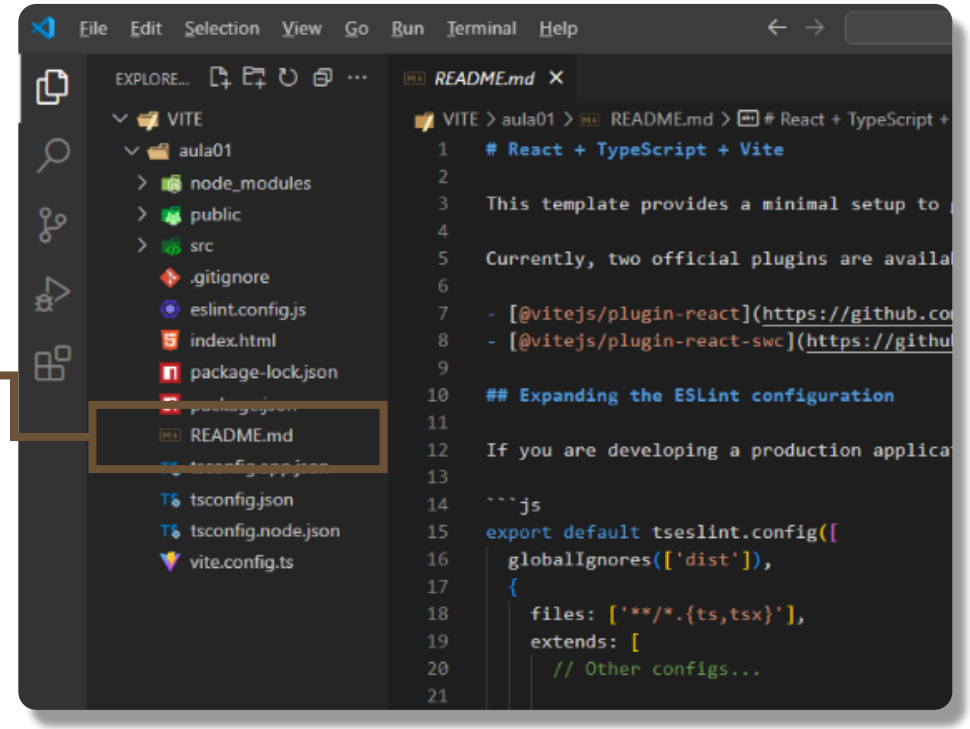
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

readme.md

É um dos arquivos mais importantes de um projeto, especialmente quando ele está hospedado em repositórios como GitHub, GitLab ou Bitbucket.

Escrito em Markdown, ele serve como uma documentação introdutória, explicando o objetivo do projeto, como instalá-lo, como utilizá-lo, requisitos, exemplos de uso, instruções de contribuição e qualquer outra informação útil.

É geralmente o primeiro contato que outros desenvolvedores terão com o projeto, funcionando como uma espécie de cartão de visita técnico, facilitando a compreensão e o engajamento com o código.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

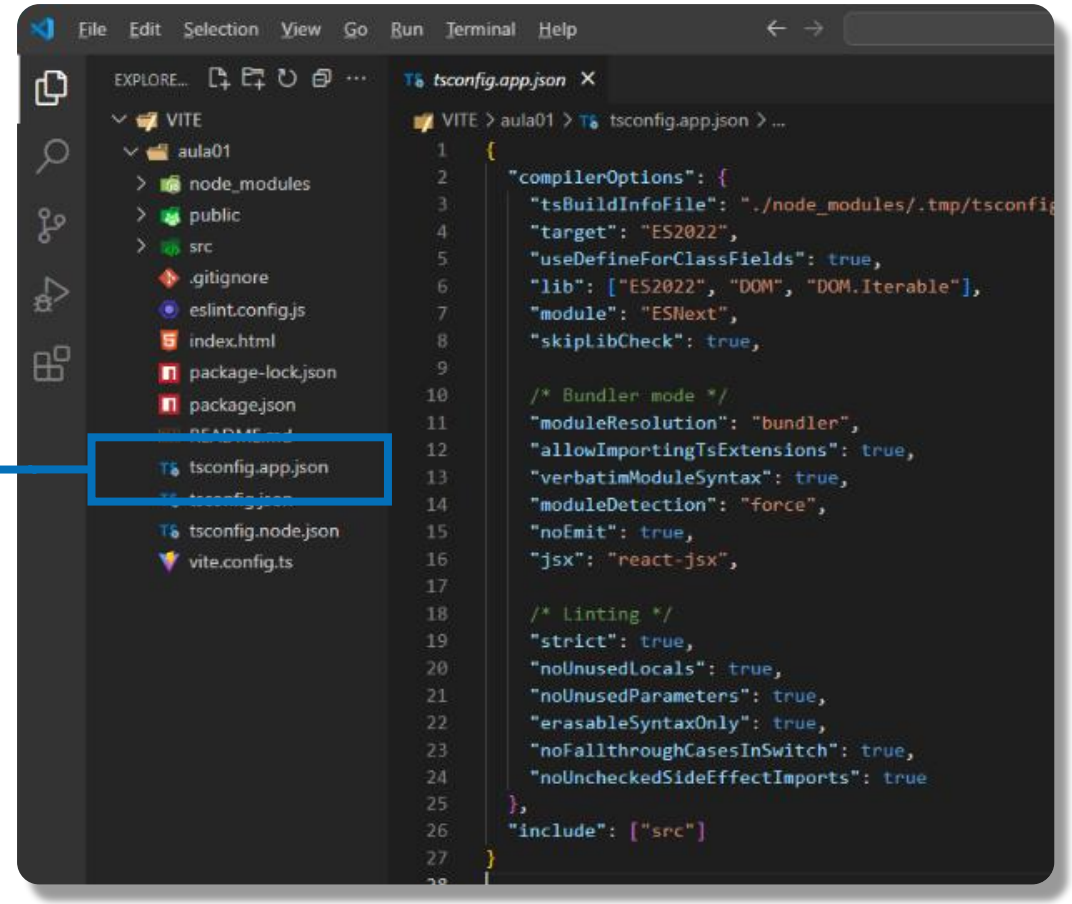
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

tsconfig.app.json

É uma configuração complementar ao tsconfig.json, usada principalmente em projetos TypeScript com estrutura modular, como em projetos Angular ou monorepos.

Define configurações específicas da aplicação, como quais arquivos devem ser incluídos na compilação (include), quais devem ser ignorados (exclude), e quais opções do compilador se aplicam somente à parte da aplicação (excluindo testes, por exemplo).

Ele geralmente é estendido pelo tsconfig.json principal, herdando suas configurações e personalizando-as conforme necessário.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

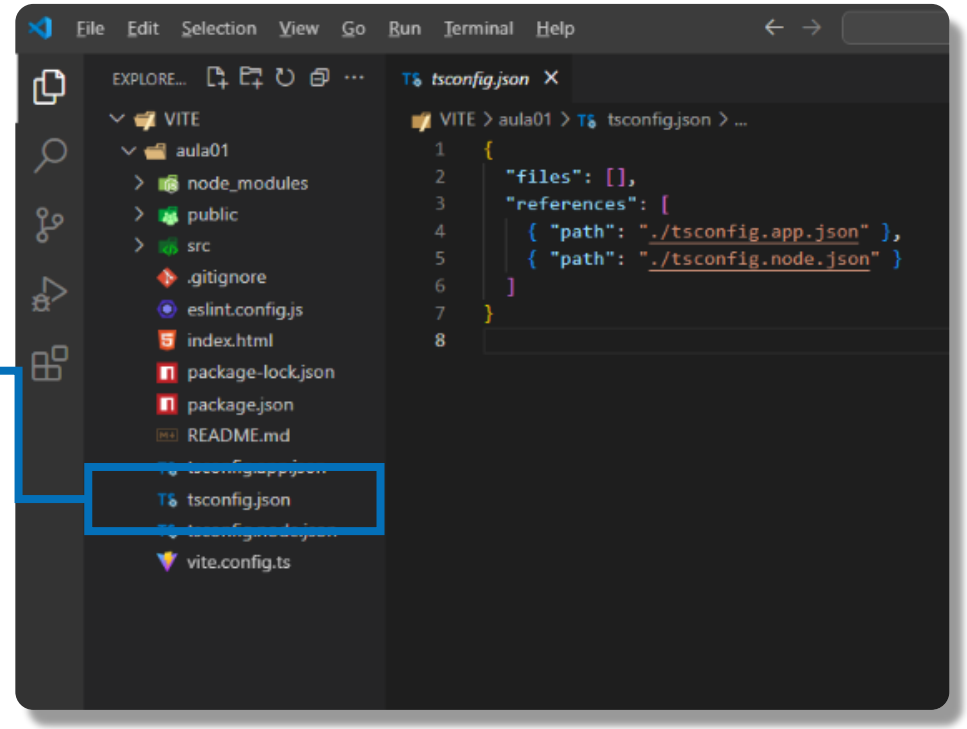
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

tsconfig.json

Configura como o compilador TypeScript deve interpretar, converter e validar o código.

Nele, definimos regras como versão de saída (target), tipo de módulo (module), verificação de tipos (strict) e quais arquivos devem ser incluídos ou ignorados.

É essencial para compilar corretamente o projeto e manter a padronização e qualidade do código.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

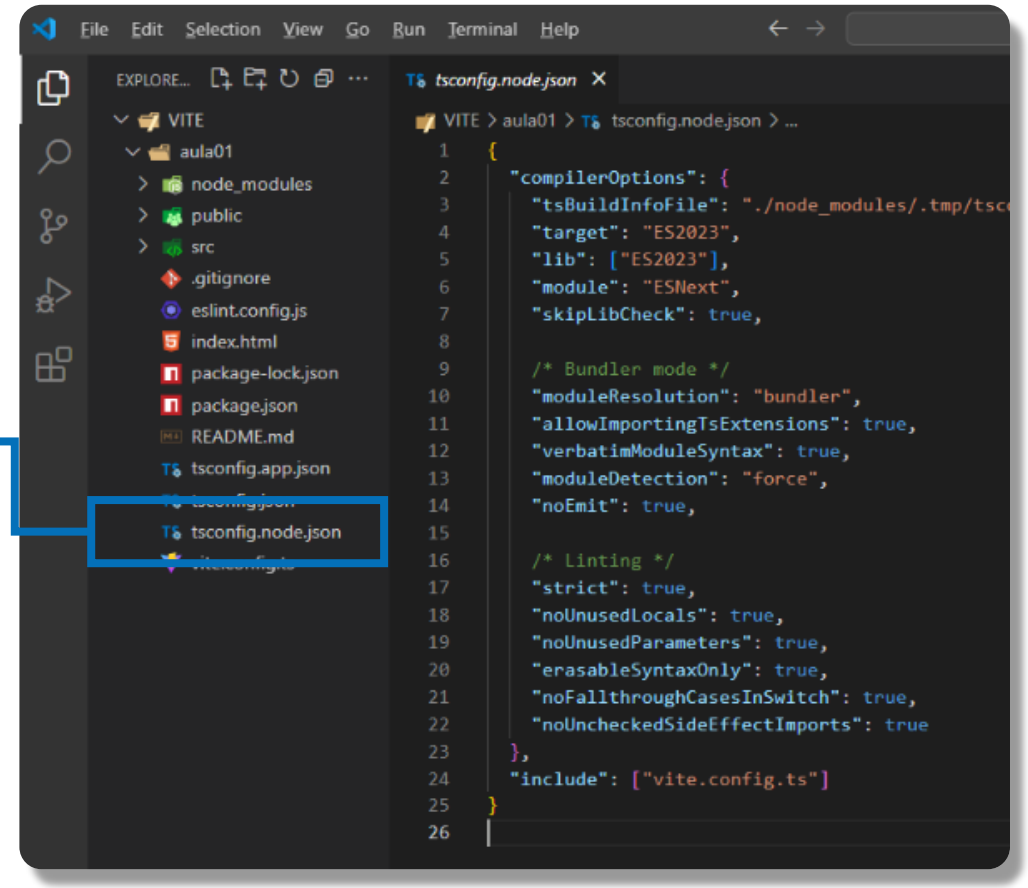
- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

tsconfig.node.json

É uma variação do tsconfig.json, criada para configurar especificamente arquivos TypeScript usados no ambiente Node.js — como scripts de build, arquivos de configuração ou scripts de servidor.

Costuma estender o tsconfig.json principal, mas ajusta opções como module, target e lib para que o código seja compatível com o Node, em vez de navegadores.

Isso permite separar as configurações da aplicação frontend das do backend ou scripts internos, mantendo o projeto mais organizado e adequado a cada ambiente.



VITE – ANATOMIA DO PROJETO INICIAL COM VITE

- ✓ Vamos analisar a estrutura de pastas e arquivos criados em nossa primeira aplicação.

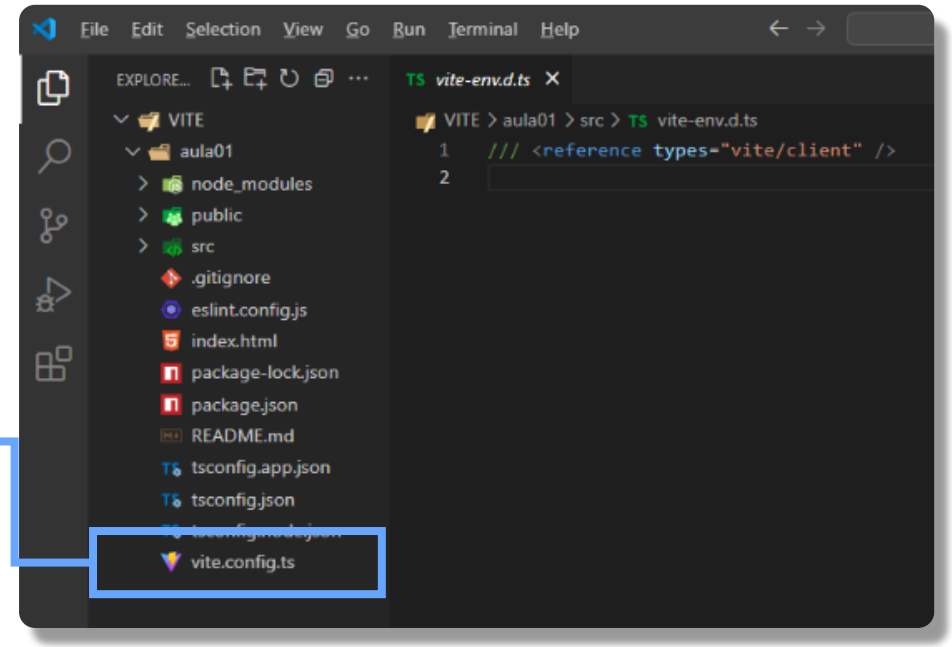
vite.config.ts

É o arquivo de configuração do Vite escrito em TypeScript.

Ele permite personalizar o comportamento da ferramenta durante o desenvolvimento e o build do projeto.

Nele, você pode configurar plugins (como React, Vue ou Tailwind), definir aliases de importação (@ para src/), modificar portas, caminhos de build, variáveis de ambiente e muito mais.

Esse arquivo é essencial quando você precisa ir além da configuração padrão do Vite, ajustando o projeto às necessidades específicas da sua aplicação.



REFERÊNCIAS BIBLIOGRÁFICAS

ABREU, Luis. Typescript - O Javascript moderno para criação de aplicações. Editora FCA, 2017.

ADRIANO, T. S. Guia prático de TypeScript. São Paulo: Casa do Código, 2021.

ANTONIO, C. Pro React: Build Complex Front-End Applications in a Composable Way With React. Apress, 2015.

BOSWELL, D; FOUCHER, T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code. Estados Unidos: O'Reilly Media, 2012.

BRITO, Robin Cris. Android Com Android Studio - Passo A Passo. Editora Ciência Moderna.

BUNA, S. React Succinctly. Estados Unidos: [s.n], 2016. Disponível em: <www.syncfusion.com/ebooks/reactjs_succinctly>. Acesso em: 12 de julho de 2025.

CHEHADE, Adib; CHEHADE, Nadia. Fullstack React with TypeScript and VITE: Build scalable and production-ready full stack web apps. 1. ed. Birmingham: Packt Publishing, 2023.

CHIN, John. VITE Quick Start Guide: Server-side rendering done right. Birmingham: Packt Publishing, 2019.

FACEBOOK (2019a). React: Getting Started. React Docs, 2019. Disponível em: <reactjs.org/docs/react-api.html>. Acesso em: 13 de julho de 2025.

FACEBOOK (2019b). React Without ES6. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-es6.html>. Acesso em: 10 de julho de 2025.

FACEBOOK (2019c). React Without JSX. React Docs, 2019. Disponível em: <reactjs.org/docs/react-without-jsx.html>. Acesso em: 10 de julho de 2025.

FREEMAN, Eric ROBSON, Elisabeth. Use a Cabeça! Programação em HTML5. Rio de Janeiro: Editora Alta Books, 2014

GACKENHEIMER, C. Introduction to React: Using React to Build scalable and efficient user interfaces.[s.i.]: Apress, 2015.

GOLDBERG, Josh. Aprendendo TypeScript: Melhore Suas Habilidades de Desenvolvimento web Usando JavaScript Type-Safe. São Paulo: Novatec. 2022

HUDSON, P. Hacking with React. 2016. Disponível em: <www.hackingwithreact.com/read/1/3/introduction-to-jsx>. Acesso em: 13 julho de 2025.

Referências Bibliográficas

KOSTRZEWA, D. Is React.js the Best JavaScript Framework in 2018? 2018. Disponível em: <hackernoon.com/is-react-js-the-best-JavaScript-framework-in-2018-264a0eb373c8>. Acesso em: 25 julho de 2025.

MARTIN, R. Clean Code: A Handbook of Agile Software Craftsmanship. Estados Unidos: Prentice Hall, 2009.

MDN WEB DOCS. Guia JavaScript. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide>. Acessado em 25 de julho de 2025.

NELSON, J. Learn React's Fundamentals Without the Buzzwords? 2018. Disponível em: <jamesknelson.com/learn-react-fundamentals-sans-buzzwords>. Acesso em: 12 julho de 2025.

VITE. The React Framework for the Web. Disponível em: https://VITE.org/. Acesso em: 23 de julho de 2025.

NIELSEN, J. Response Times: The 3 Important Limits. 1993. Disponível em: <www.nngroup.com/articles/response-times-3-important-limits>. Acesso em: 10 julho de 2025.

O'REILLY, T. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <www.oreilly.com/pub/a/web2/archive/what-is-web-20.html#mememap>. Acesso em: 10 de julho de 2025.

PANDIT, N. What Is ReactJS and Why Should We Use It? 2018. Disponível em: <www.c-sharpcorner.com/article/what-and-why-reactjs>. Acesso em: 12 de julho de 2025.

RAUSCHMAYER, A. Speaking JavaScript: An In-Depth Guide for Programmers. Estados Unidos: O'Reilly Media, 2014.

REACTIVA. O arquivo package-lock.json. Disponível em: <https://nodejs.reativa.dev/0020-package-lock-json/index>. Acessado em 13 de julho de 2025.

_____. O guia do package.json. Disponível em: <https://nodejs.reativa.dev/0019-package-json/index>. Acessado em 13 de julho de 2025.

RICOY, L. Desmitificando React: Uma Reflexão para Iniciantes. 2018. Disponível em: <medium.com/trainingcenter/desmitificando-react-uma-reflex%C3%A3o-para-iniciantes-a57af90b6114>. Acesso em: 13 julho de 2025.

SHARMA, Nabendu Biswas; UDAYAN, Kuldeep. Building Microfrontends with React 18 and VITE 13: A complete guide to building modern microfrontend-based web apps using React and VITE. 1. ed. Birmingham: Packt Publishing, 2023.

Referências Bibliográficas

SILVA, Maurício Samy. Ajax com jQuery: requisições Ajax com a simplicidade de jQuery. São Paulo: Novatec Editora, 2009.

_____. Construindo Sites com CSS e XHTML. Sites Controlados por Folhas de Estilo em Cascata. São Paulo: Novatec, 2010.

_____. CSS3 - Desenvolva aplicações web profissionais com o uso dos poderosos recursos de estilização das CSS. São Paulo: Novatec Editora, 2010.

STACKOVERFLOW. Most Popular Technologies: Web Frameworks. Developer Survey Results, StackOverflow, 2019. Disponível em: <insights.stackoverflow.com/survey/2019#technology>. Acesso em: 13 de julho de 2025.

STATE OF JS 2024. State of JavaScript 2024. Disponível em: <https://2024.stateofjs.com/en-US/>. Acesso em: 03 de julho de2025.

SWC. Rust-based platform for the Web. 2024 Disponível em <https://swc.rs/>. Acessado em 23 de julho de 2025.

VERCEL, Inc. VITE: The React Framework. Disponível em: https://VITE.org/. Acesso em: 25 de julho de 2025.

VERCEL, Inc. VITE Documentation. Disponível em: https://VITE.org/docs. Acesso em: 25 de julho de 2025.

W3C. HTML5 - A linguagem de marcação que revolucionou a web. São Paulo: Novatec Editora, 2010.

_____. A vocabulary and associated APIs for HTML and XHTML. Disponível em <https://www.w3.org/TR/2018/SPSD-html5-20180327/>. Acessado em 22 de julho de 2025.

_____. Cascading Style Sheets, level 1. Disponível em <https://www.w3.org/TR/2018/SPSD-CSS1-20180913/>. Acessado em 22 de julho de 2025.

_____. Cascading Style Sheets, level 2 Revision 2. Disponível em <https://www.w3.org/TR/2016/WD-CSS22-20160412/>. Acessado em 22 de julho de 2025.

_____. Cascading Style Sheets, level 2. Disponível em <https://www.w3.org/TR/2008/REC-CSS2-20080411/>. Acessado em 22 de julho de 2025.

_____. Cascading Style Sheets, level 3. Disponível em <https://www.w3.org/TR/css-syntax-3/>. Acessado em 22 de julho de 2025.

REFERÊNCIAS BIBLIOGRÁFICAS

W3C. HTML 3.2 Reference Specification. Disponível em <https://www.w3.org/TR/2018/SPSD-html32-20180315/>. Acessado em 20 de julho de 2025.

_____. HTML 4.0 Specification. Disponível em <https://www.w3.org/TR/1998/REC-html40-19980424/>. Acessado em 20 de julho de 2025.

_____. HTML 4.01 Specification. Disponível em <https://www.w3.org/TR/2018/SPSD-html401-20180327/>. Acessado em 20 de julho de 2025.

_____.Cascading Style Sheets, level 2 Revision 1. Disponível em <https://www.w3.org/TR/CSS2/>. Acessado em 20 de julho de 2025.

WIKIPEDIA. JavaScript. Disponível em <https://pt.wikipedia.org/wiki/JavaScript>. Acessado em 20 de julho de 2025.

DÚVIDAS?

CRÍTICAS?

SUGESTÕES?

AMEAÇAS?